

# Three dimensional thermal-solute phase field simulation of binary alloy solidification

P.C. Bollada\*, C.E. Goodyer†, P.K. Jimack, A.M. Mullis, F.W. Yang

September 23, 2014

## Abstract

We employ adaptive mesh refinement, implicit time stepping, a nonlinear multigrid solver and parallel computation, to solve a multi-scale, time dependent, three dimensional, nonlinear set of coupled partial differential equations for three scalar field variables. The mathematical model represents the non-isothermal solidification of a metal alloy into a melt substantially cooled below its freezing point at the microscale. Underlying physical molecular forces are captured at this scale by a specification of the energy field. The time rate of change of the temperature, alloy concentration and an order parameter to govern the state of the material (liquid or solid) is controlled by the diffusion parameters and variational derivatives of the energy functional. The physical problem is important to material scientists for the development of solid metal alloys and, hitherto, this fully coupled thermal problem has not been simulated in three dimensions, due to its computationally demanding nature. By bringing together state of the art numerical techniques this problem is now shown here to be tractable at appropriate resolution with relatively moderate computational resources.

## 1 Introduction

We here present our computational approach to simulating, at the meso scale, three dimensional, non-isothermal, alloy solidification from an initial small, spherical seed into a mature, dendritic crystal. A feature of a mature dendrite is the geometric complexity of its evolving two-dimensional surface (see Fig. 1 for a typical snapshot in time). This makes tracking of the surface a difficult task in sharp interface models. A phase-field model avoids this by making use of the phase field,  $\phi(\mathbf{x}, t) \in [-1, 1]$ , to represent, at its two extremes, the liquid and solid state respectively and the evolution of the phase boundary,  $\phi = 0$ , is the surface of interest. This solves one problem, but at a cost of introducing another. The computation requires an extra variable, the phase, which varies rapidly over a small region about the interface. Taking the thickness of the interface to be  $\approx 1$ , we find the size of a mature dendrite grows to  $\sim 300$ , requiring the domain size to be significantly greater still (depending on the thermal field this may need to be  $O(1000)$ , or even more), we see that the interface region of interest is very much smaller than the overall domain. Of major concern in phase field models is the dependence of the computed results on the interface width. To address this, Karma [1], analysed the problem in the thin interface limit to produce a phase field formulation that is independent of the interface width up to several orders of magnitude larger than a physically real value, although this is less clear when a thermal field is coupled. That said, we adopt an interface width that is of physically realistic order. There are two coupled driving forces for growth: the alloy concentration, governed by a diffusion parameter  $D_c$  and, secondly, a temperature field governed by a diffusion parameter  $D_\theta$ . The ratio of these two parameters gives the Lewis number,  $Le = D_\theta/D_c$ , which for many metallic alloys approaches 10,000. In two dimensions Lewis numbers of this magnitude have been realised by [2]. However, there are no prior results, even for the interface widths permitted by Karma's model, for even very moderate Lewis number in three dimensions. This paper seeks to demonstrate that such results are feasible provided the appropriate numerical techniques are employed.

The numerical solution to this phase field model (described in detail in the following section) requires methods to solve a time-dependent, highly nonlinear system of PDEs, of parabolic type, and capable of resolving varying length and time scales. A feature that adds another level of difficulty to this problem is that we are particularly interested in the tip radius and speed of growth of the dendrite only when it is fully mature, and these two numbers are steady. In summary, the computational problem is: non-linear, three-dimensional, stiff, involves multiple length scales to capture small phase and large temperature fields, multi-time scale associated with the Lewis number and, to establish a mature dendrite, requires a long simulation time.

\*p.c.bollada@leeds.ac.uk

†Now at: Numerical Algorithms Group: all other authors at University of Leeds U.K.

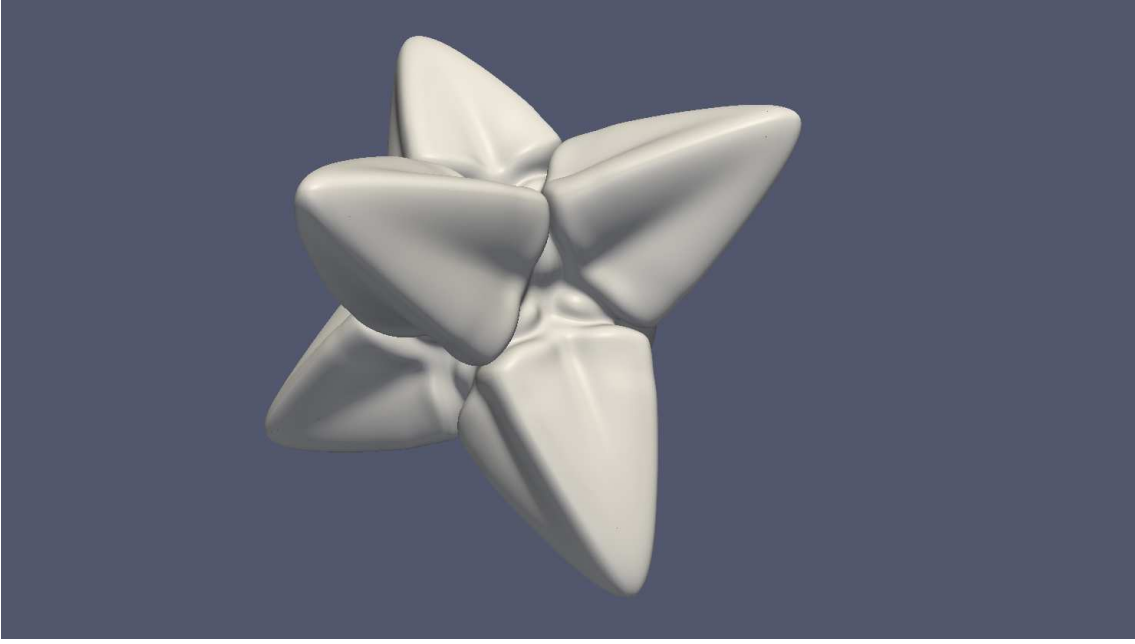


Figure 1: Snapshot of the solid-liquid interface for a typical dendrite. This image was obtained from a simulation with  $Le = 40$ ,  $\Delta = 0.525$  and  $\Delta x = 0.78$ .

The computational techniques we employ are: use of very fine meshing in the region around the moving boundary where phase field and solute field resolution is critical, and coarse meshing away from the boundary where only the slowly changing temperature field requires resolution; implicit time stepping to allow much larger time steps than would otherwise be possible; nonlinear smoothing in conjunction with a nonlinear multi-grid solver; and parallel processing with up to 1024 cores as the simulation progresses. The combination of all of these techniques allows an almost optimal solution process to be developed, in which the number of degrees of freedom is evolved with the dendrite, to maintain the required resolution as the interface grows, and the solution time at each time step is approximately proportional to the number of degrees of freedom. Furthermore, the use of a parallel implementation ensures that sufficient primary memory is available to support a mesh resolution which is fully converged whilst maintaining a tractable solution time.

The particular phase field model we employ is an extension of [3], and is based on the three dimensional thermal-phase field model of [4] and two dimensional thermal-solutal phase field model of [5]. One feature of the physical problem is that it is purely dissipative, or entropy increasing, as all natural relaxational phenomena are. The resulting PDEs are of Allen-Cahn [6] and Cahn-Hilliard type [7]. That is to say, the model involves time derivatives of the three fields coupled to forms involving variational derivatives of some functional - typically the free energy functional. As the dendrite grows the free energy reduces monotonically with time but never achieves equilibrium if the domain boundary is far from the dendrite. Although we have listed some of the difficult aspects of this model, the relaxational aspect is typically an asset and results in stable numerical schemes: there is no convection, for example (at least in the absence of flow in the melt).

The variational form of the mathematical model is, of course, identical to the two-dimensional model in form. However, on realising the variational derivatives the resulting equations are more complex and nonlinear in the higher dimension. This is largely because of surface energy related anisotropy associated to alignment at the molecular scale. In two dimensions anisotropy is conveniently formulated using a single angle parameter, but in three dimensions we prefer to use a normal given in terms of Cartesian gradients.

In addition to the Lewis number, another key parameter in the simulations is the undercooling,  $\Delta$ , which sets the temperature of the liquid's initial and far boundary condition below its freezing point. As this parameter becomes larger the under-cooling becomes more severe, the solification more rapid and fractal in appearance: and, also, correspondingly more difficult to simulate.

The other field, not hitherto discussed, is the solute field. For a binary alloy of initial concentration,  $c_\infty$ , the concentration of an alloy component at any point is represented by a value of  $c(\mathbf{x}, t) \in [0, 1]$ . The requirement for equilibrium at the solid-liquid interface means that the concentration in the solid and the concentration in the liquid at the interface will be unequal. In a sharp interface model this results in a discontinuous jump in  $c$  at the interface, while in phase field models it results in a steep, but continuous, increase in  $c$  across the diffuse interface region, where there is some advantage in reformulating the problem to remove this. We show this in the next section.

## 2 Governing equations

The governing equations for dendritic growth of an under-cooled binary alloy are here presented in full, in both their variational form and in the (equivalent) form of PDEs for numerical implementation. The non-dimensional equations for the phase field,  $\phi$ , the solute concentration,  $c$  and the dimensionless temperature,  $\theta$ , are given via a specification of the free energy

$$F \equiv \int_V \frac{1}{2} A(\underline{n})^2 \nabla \phi \cdot \nabla \phi + f(\theta, \phi) dV \quad (1)$$

and the relations

$$\tau(c, \phi) A^2(\underline{n}) \dot{\phi} = - \frac{\delta F}{\delta \phi} \quad (2a)$$

$$\dot{c} = \nabla \cdot \left( K(\phi) \nabla \frac{\delta F}{\delta c} - \underline{j} \right), \quad (2b)$$

$$\dot{\theta} = D_\theta \nabla^2 \theta + \frac{1}{2} \dot{\phi}. \quad (2c)$$

The solute diffusion parameter is given by

$$K = D_c \frac{1}{2} (1 - \phi). \quad (3)$$

The parameter  $D_c$  is a diffusion constant, thus  $K = 0$  in the solid ( $\phi = 1$ ) and  $K = D_c$  in the liquid ( $\phi = -1$ ).  $D_\theta$  is the temperature diffusion coefficient (assumed constant). The normal to the interface is given by

$$\underline{n} = \frac{\nabla \phi}{|\nabla \phi|}, \quad (4)$$

which is well defined around  $\phi = 0$ , and the anisotropy function for cubic symmetry (growth is preferred along the normals to the faces) is given for three dimensions by [4],

$$A(\underline{n}) \equiv A_0 [1 + \tilde{\epsilon} (n_x^4 + n_y^4 + n_z^4)] \quad (5)$$

where  $\underline{n} = [n_x, n_y, n_z]$ ,  $A_0 = 1 - 3\epsilon$ ,  $\tilde{\epsilon} = 4\epsilon/(1 - 3\epsilon)$  and  $\epsilon \approx 0.02$  governs the amount of anisotropy. The reason for this arrangement of constants is to compare with the two dimensional form

$$A(\underline{n}) \equiv A_0 [1 + \tilde{\epsilon} (n_x^4 + n_y^4)] \equiv 1 + \epsilon \cos 4\psi \quad (6)$$

where the angle,  $\psi$ , is given by

$$\tan \psi = \frac{\partial \phi}{\partial y} / \frac{\partial \phi}{\partial x}. \quad (7)$$

The dimensionless relaxation time function is defined by

$$\tau(c, \phi) \equiv \frac{1}{Le} + Mc_\infty [1 + (1 - k_E)U], \quad (8)$$

where the Lewis number  $Le = D_\theta/D_c$  and

$$U \equiv \frac{1}{1 - k_E} \left( \frac{2c/c_\infty}{1 + k_E - (1 - k_E)\phi} - 1 \right). \quad (9)$$

Here  $k_E$  is the equilibrium partition coefficient,  $c_\infty$  is the far boundary condition for  $c$ . The anti-trapping current  $\underline{j}$ , appearing in the solute equation, Eq. 2b, is prescribed by

$$\underline{j} = - \frac{1}{2\sqrt{2}} [1 + (1 - k_E)] U \dot{\phi} \underline{n}, \quad (10)$$

The profile of  $c$  exhibits a spike at the interface, which can present computational difficulties. Following [5], this is largely overcome by rewriting the solute equation using the variable  $U$ :

$$\left( \frac{1 + k_E}{2} - \frac{1 - k_E}{2} \phi \right) \frac{\partial U}{\partial t} = \nabla \cdot \left\{ D_c \frac{1 - \phi}{2} \nabla U + \underline{j} \right\} + \frac{1}{2} [1 + (1 - k_E)U] \frac{\partial \phi}{\partial t}. \quad (11)$$

The physical temperature field,  $T$ , can be recovered by the relationship

$$\theta = \frac{T - T_M - mc_\infty}{L/C_p}, \quad (12)$$

where  $L$  and  $C_p$  are the latent heat of the phase transition and heat capacity respectively. The slope of the liquidus line is given by  $m = ML/[C_p(1 - \kappa_E)]$  and  $T_M$  is the melting temperature of the alloy.

Finally the bulk free energy density is given by

$$f(\theta, \phi) \equiv \frac{\phi^2}{2} \left( \frac{\phi^2}{2} - 1 \right) + \lambda(\theta + c_\infty U) \left( \phi - \frac{2\phi^3}{3} + \frac{\phi^5}{5} \right). \quad (13)$$

We solve the system of equations 2a, 2c and 11 plus initial (typically small) solid seed see subsection 3.2 and far boundary conditions

$$\begin{aligned} \phi|_{\text{far}} &= -1 \\ U|_{\text{far}} &= 0 \quad (\equiv c|_{\text{far}} = c_\infty) \\ \theta|_{\text{far}} &= -\Delta \end{aligned} \quad (14)$$

where  $\Delta$  is the given under-cooling. The equation for temperature is a standard diffusion equation<sup>1</sup> with a heating term,  $\dot{\phi}$ , proportional to the solidification rate (or cooling if melting). The driving force for the phase equations is given by  $f(T, \phi)$ , consisting of a double well potential having stable minima at  $\phi = \pm 1$  and a maximum at  $\phi = 0$  and a function of  $\theta$  to create conditions for moving the phase boundary. For example a negative value of  $\theta$  creates conditions favourable for solidification. The parameter,  $\lambda$ , is proportional to the interface width, which in turn is chosen as the characteristic length scale.

## 2.1 Parameter values

For the purposes of this paper we choose a selection of parameters to use as default values for the simulations below in Table 1. Any deviation from these parameter values is explicitly noted in the text.

Physical property	Symbol	value
Anisotropy	$\epsilon$	0.02
Boundary concentration	$Mc_\infty$	0.05
Equilibrium partition coefficient	$\kappa_E$	0.3
Dimensionless interface width	$\lambda$	2
Ratio of solute diffusivity to characteristic diffusivity	$D_c$	1.2534
Lewis number - $D_\theta/D_c$	Le	40 and 100
Dimensionless Undercooling at the far boundary	$\Delta$	0.25 to 0.80
Initial nuclear radius	$R_0$	5.0
Computational property	symbol	value
Finest grid size	$\Delta x$	0.195 to 0.78
Computation domain size	$L^3$	800×800×800

Table 1: Table of parameter values used for the simulations in this paper.

## 2.2 Anisotropic calculations

Note that the phase equation, Eq. 2a, is made considerably more complicated by the presence of the anisotropy term,  $A(\underline{n})$ , in the free energy functional. The variational derivative of a functional not involving gradients is simply the partial derivative of the density with respect to that variable. Thus

$$\begin{aligned} \frac{\delta}{\delta\phi} \int f(T, \phi) dV &= \frac{\partial f}{\partial\phi} \\ &= \phi^3 - \phi + \lambda(\theta + c_\infty U)(1 - 2\phi^2 + \phi^4). \end{aligned} \quad (15)$$

The variational derivative of the pure gradient part of the functional is given by

$$\frac{\delta G}{\delta\phi} \equiv \frac{\delta}{\delta\phi} \int g(\nabla\phi) dV = -\nabla \cdot \left( \frac{\partial}{\partial\mathbf{r}} g(\mathbf{r}) \Big|_{\mathbf{r}=\nabla\phi} \right) \quad (16)$$

<sup>1</sup>without convection due to no velocity field

where, in our model,

$$g(\mathbf{r}) \equiv \frac{1}{2}A(\mathbf{n})^2|\mathbf{r}|^2, \text{ for } \mathbf{r} \in \mathbb{R}^3 \quad (17)$$

In order to expand Eq. 16 and thus, Eq. 2a, we first introduce the notation:  $\phi_{,i} \equiv \partial_i \phi \equiv \frac{\partial \phi}{\partial x^i}$  etc., for Cartesian differentiation, and subscripts for differentiation on function space. Thus

$$\begin{aligned} g_i &\equiv \frac{\partial g}{\partial r_i}, \\ g_{ij} &\equiv \frac{\partial}{\partial r_i} \frac{\partial g}{\partial r_j}. \end{aligned} \quad (18)$$

This enables us to write

$$\begin{aligned} -\frac{\delta G}{\delta \phi} &= \partial_i g_i \\ &= \phi_{,ij} g_{ij} \quad (\text{using the chain rule}), \end{aligned}$$

which written out in full reads

$$-\frac{\delta G}{\delta \phi} \equiv -\frac{\delta}{\delta \phi} \int g(\nabla \phi) dx^3 = \frac{\partial^2 \phi}{\partial x^i \partial x^j} \left( \frac{\partial}{\partial r_i} \frac{\partial g}{\partial r_j} \right) \Big|_{\mathbf{r}=\nabla \phi}.$$

Note that,  $g_{ij}$ , is a function of only first derivatives of  $\phi$ . To avoid expanding the above in terms of the components,  $\phi_{,i}$  we first introduce the substitutions  $q \equiv |\nabla \phi|^2$  and  $\mathbf{X} \equiv [X_1, X_2, X_3] \equiv [\phi_{,1}^2/q, \phi_{,2}^2/q, \phi_{,3}^2/q]$ , to write the anisotropy

$$A = A_0 \left( 1 + \tilde{\epsilon} \sum_{i=1}^3 X_i^2 \right). \quad (19)$$

For an arbitrary function  $h(\mathbf{r}) = \tilde{h}(\mathbf{r}, q, \mathbf{X}, A)$  we use the chain rule to write

$$\begin{aligned} \frac{\partial h}{\partial r_i} &= \left( \frac{\partial}{\partial r_i} + \frac{\partial q}{\partial r_i} \frac{\partial}{\partial q} + \frac{\partial X_j}{\partial r_i} \frac{\partial}{\partial X_j} + \frac{\partial A}{\partial r_i} \frac{\partial}{\partial A} \right) \tilde{h} \\ &= \left( \frac{\partial}{\partial r_i} + 2r_i \frac{\partial}{\partial q} + \frac{2r_i}{q} (\delta_{ij} - X_j) \frac{\partial}{\partial X_j} + \frac{2r_i}{q} [2A_0 \tilde{\epsilon} X_i - 2(A - A_0)] \frac{\partial}{\partial A} \right) \tilde{h}, \end{aligned} \quad (20)$$

where we have used

$$\begin{aligned} \frac{\partial q}{\partial r_i} &= 2r_i, \\ \frac{\partial X_j}{\partial r_i} &\equiv \frac{\partial}{\partial r_i} \left( \frac{r_j^2}{q} \right) = \frac{2r_i \delta_{ij}}{q} - \frac{2r_i r_j^2}{q^2} = \frac{2r_i}{q} (\delta_{ij} - X_j), \\ \frac{\partial A}{\partial r_i} &= \frac{\partial X_j}{\partial r_i} \frac{\partial A}{\partial X_j} = \frac{2r_i}{q} (\delta_{ij} - X_j) \frac{\partial A}{\partial X_j} \\ &= \frac{2r_i}{q} (\delta_{ij} - X_j) 2A_0 \tilde{\epsilon} X_j \\ &= \frac{4r_i}{q} (A_0 \tilde{\epsilon} X_i - A + A_0) \end{aligned} \quad (21)$$

using, on the last simplification, the identity  $A - A_0 \equiv A_0 \tilde{\epsilon} \sum_j X_j^2$ . This allows us to compute

$$g_i \equiv \frac{\partial}{\partial r_i} \left( \frac{1}{2} A^2 q \right) \Big|_{\mathbf{r}=\nabla \phi} = \phi_{,i} A^2 + 4\phi_{,i} (A_0 X_i \tilde{\epsilon} - A + A_0) A \quad (22)$$

and further differentiation gives the concise forms

$$\begin{aligned} g_{ii} &= (24X_i - 3)A^2 + (-48X_i^2 \tilde{\epsilon} + 12X_i \tilde{\epsilon} - 40X_i + 4)A_0 A + 16X_i (X_i \tilde{\epsilon} + 1)^2 A_0^2 \\ g_{ij} &= \frac{\phi_{,i} \phi_{,j}}{g} [24A^2 + (-24X_i \tilde{\epsilon} - 24X_j \tilde{\epsilon} - 40)A_0 A + (16(X_j \tilde{\epsilon} + 1)(X_i \tilde{\epsilon} + 1)A_0^2)], \quad i \neq j. \end{aligned} \quad (23)$$

The above expressions are not only much more concise than the expanded equivalent as a function of  $\phi_{,i}$ , but are functions of  $X_i$  and  $A$  which are of order unity in size and thus minimise floating point errors (the expanded equivalent contains tenth order polynomials of  $\phi_{,i}$ ). We note also that, in the absence of anisotropy,  $\tilde{\epsilon} = 0$ , reduces  $g_{ij}$  to  $\delta_{ij}$  so that  $\phi_{,ij}g_{ij} = \nabla^2\phi$ . In the situation where  $g_{ij}$  is ill defined due to  $|\nabla\phi| \rightarrow 0$  we set.

$$\phi_{,ij}g_{ij}|_{|\nabla\phi|\rightarrow 0} = A_0^2(1 + \tilde{\epsilon})^2\nabla^2\phi \quad (24)$$

or, equivalently  $A|_{|\nabla\phi|\rightarrow 0} = A_0(1 + \tilde{\epsilon})$ . In practice we use this expression only when  $|\nabla\phi| = 0$ , to machine precision without difficulty.

Using the notation  $\text{tr}(\mathbf{g}) \equiv \delta_{ij}g_{ij}$ , the rearrangement

$$\begin{aligned} \phi_{,ij}g_{ij} &\equiv \frac{1}{3}(\phi_{,11} + \phi_{,22} + \phi_{,33})(g_{11} + g_{22} + g_{33}) + (\phi_{,ij} - \frac{1}{3}\phi_{,kk}\delta_{ij})g_{ij} \\ &\equiv \frac{1}{3}\nabla^2\phi \text{tr}(\mathbf{g}) + (\phi_{,ij} - \frac{1}{3}\phi_{,kk}\delta_{ij})g_{ij}, \end{aligned} \quad (25)$$

allows the dominant term to be isolated and has advantage, because the Laplacian can be discretised to minimise grid induced anisotropy. We will also see that the term  $(\phi_{,ij} - \frac{1}{3}\phi_{,kk}\delta_{ij})$ , like  $g_{ij}$ , on discretisation with a compact stencil at a discrete node,  $\mathbf{p}$ , only has contributions from the nodes surrounding  $\mathbf{p}$ . This affords simplification for the non-linear solver later discussed.

### 2.3 System summary

Writing,  $M_{ij} \equiv \phi_{,ij} - \frac{1}{3}\phi_{,kk}\delta_{ij}$  we summarise the nonlinear PDE system that forms our mathematical model as

$$\tau(c, \phi)A(\mathbf{n})^2\dot{\phi} = \frac{1}{3}\nabla^2\phi \text{tr}(\mathbf{g}) + M_{ij}g_{ij} - \frac{\partial f}{\partial \phi} \quad (26)$$

where  $\tau(c, \phi)$  is given by Eq. 8,  $\frac{\partial f}{\partial \phi}$  is given in Eq. 15,  $g_{ij}$  in Eq. 23, the solute is solved via Eq. 11 and the temperature by Eq. 2c.

## 3 Discretisation

The approach taken to discretisation is based upon a cell centred finite difference scheme, in that the nodes of the domain are located at the centre of cubic cells. and thus, we use the term ‘node’ and ‘cell centre’ interchangeably. One consequence of this is that there are no nodes on the domain boundary, thus making the use of Dirichlet boundary conditions non-trivial. The scheme makes use of the PARAMESH library to support mesh adaptivity in parallel [8, 9]. The meshes obtained by this approach take the form of an oct tree of regular blocks, within which the mesh is uniform, and it is the spatial discretisation on any one of these blocks that we discuss here. Subsequently we will discuss adaptive mesh refinement and the implicit temporal discretisation scheme that is employed.

### 3.1 Spatial discretisation

Compact finite difference stencils ( $3 \times 3 \times 3$ ), are used to discretise the first and second derivatives. Denoting these 27 points by  $\mathbf{Q}$  and defining a generic 27 point Laplacian stencil,  $W_{abc}$ , around a point  $\mathbf{p} = [i, j, k]$  by

$$\nabla^2 u|_{\mathbf{Q}} = \frac{1}{(\Delta x)^2} \sum_{a=-1}^1 \sum_{b=-1}^1 \sum_{c=-1}^1 W_{abc} u_{\mathbf{p}+[a,b,c]} \quad (27)$$

where  $\Delta x$  is the physical distance between nearest neighbours, we can recover the 7 point Laplacian stencil, built from only the centre node,  $\mathbf{p}$  and the 6 nearest neighbours ( $a^2 + b^2 + c^2 = 1$ )

$$\nabla^2 u|_{\mathbf{Q}} = \frac{-6u_{i,j,k} + u_{i+1,j,k} + u_{i-1,j,k} + u_{i,j+1,k} + u_{i,j-1,k} + u_{i,j,k+1} + u_{i,j,k-1}}{(\Delta x)^2} \quad (28)$$

by setting the weights

$$W_{abc} = \begin{cases} -6 & a^2 + b^2 + c^2 = 0 \\ 1 & a^2 + b^2 + c^2 = 1 \\ 0 & \text{otherwise} \end{cases} \quad (29)$$

However, this stencil is more prone to grid anisotropy than the following 27 point Laplacian stencil (see [10]), with weights

$$W_{abc} = \begin{cases} -128/30 & a^2 + b^2 + c^2 = 0 \\ 14/30 & a^2 + b^2 + c^2 = 1 \\ 3/30 & a^2 + b^2 + c^2 = 2 \\ 1/30 & a^2 + b^2 + c^2 = 3 \end{cases} \quad (30)$$

In order to discretise the phase equation, Eq.26, in space it is necessary to approximate  $\phi_{,ij}g_{ij}$  about the point  $\mathbf{p}$ . Using the above notation we obtain:

$$(\phi_{,ij}g_{ij})|_{\mathbf{Q}} = \frac{1}{3}\nabla^2\phi|_{\mathbf{Q}}\text{tr}(\mathbf{g})|_{(\mathbf{Q}-\mathbf{p})} + M_{ij}|_{(\mathbf{Q}-\mathbf{p})}g_{ij}|_{(\mathbf{Q}-\mathbf{p})}, \quad (31)$$

where we use the notation,  $|_{\mathbf{Q}-\mathbf{p}}$ , to denote that the central node is not used. We discretise,  $M_{ij}$  as follows

$$\begin{aligned} \Delta|_{\mathbf{Q}-\mathbf{p}} &\equiv \phi_{\mathbf{p}+[1,0,0]} + \phi_{\mathbf{p}+[-1,0,0]} + \phi_{\mathbf{p}+[0,1,0]} + \phi_{\mathbf{p}+[0,-1,0]} + \phi_{\mathbf{p}+[0,0,1]} + \phi_{\mathbf{p}+[0,0,-1]} \\ M_{11}|_{\mathbf{Q}-\mathbf{p}} &= \frac{1}{\Delta x^2} (\phi_{\mathbf{p}+[1,0,0]} + \phi_{\mathbf{p}+[-1,0,0]} - \frac{1}{3}\Delta|_{\mathbf{Q}-\mathbf{p}}), \\ M_{22}|_{\mathbf{Q}-\mathbf{p}} &= \frac{1}{\Delta x^2} (\phi_{\mathbf{p}+[0,1,0]} + \phi_{\mathbf{p}+[0,-1,0]} - \frac{1}{3}\Delta|_{\mathbf{Q}-\mathbf{p}}), \\ M_{33}|_{\mathbf{Q}-\mathbf{p}} &= \frac{1}{\Delta x^2} (\phi_{\mathbf{p}+[0,0,1]} + \phi_{\mathbf{p}+[0,0,-1]} - \frac{1}{3}\Delta|_{\mathbf{Q}-\mathbf{p}}), \\ M_{12}|_{\mathbf{Q}-\mathbf{p}} = M_{21}|_{\mathbf{Q}-\mathbf{p}} &= \frac{1}{4\Delta x^2} (\phi_{\mathbf{p}+[1,1,0]} + \phi_{\mathbf{p}+[-1,-1,0]} - \phi_{\mathbf{p}+[1,-1,0]} - \phi_{\mathbf{p}+[-1,1,0]}) \\ M_{23}|_{\mathbf{Q}-\mathbf{p}} = M_{32}|_{\mathbf{Q}-\mathbf{p}} &= \frac{1}{4\Delta x^2} (\phi_{\mathbf{p}+[0,1,1]} + \phi_{\mathbf{p}+[0,-1,-1]} - \phi_{\mathbf{p}+[0,-1,1]} - \phi_{\mathbf{p}+[0,1,-1]}) \\ M_{31}|_{\mathbf{Q}-\mathbf{p}} = M_{13}|_{\mathbf{Q}-\mathbf{p}} &= \frac{1}{4\Delta x^2} (\phi_{\mathbf{p}+[1,0,1]} + \phi_{\mathbf{p}+[-1,0,-1]} - \phi_{\mathbf{p}+[1,0,-1]} - \phi_{\mathbf{p}+[-1,0,1]}), \end{aligned} \quad (32)$$

where  $|_{\mathbf{Q}-\mathbf{p}}$  denotes use of some or all of the  $3^3 - 1$  surrounding nodes,  $\mathbf{p} + [1, 0, 0], \mathbf{p} + [0, 1, 0], \dots, \mathbf{p} + [1, 1, 1]$ . The matrix elements,  $g_{ij}$ , are functions of the components of  $\nabla\phi$  only:

$$\begin{aligned} \phi_{,1}|_{\mathbf{Q}-\mathbf{p}} &= \frac{1}{2\Delta x} (\phi_{\mathbf{p}+[1,0,0]} - \phi_{\mathbf{p}+[-1,0,0]}) \\ \phi_{,2}|_{\mathbf{Q}-\mathbf{p}} &= \frac{1}{2\Delta x} (\phi_{\mathbf{p}+[0,1,0]} - \phi_{\mathbf{p}+[0,-1,0]}) \\ \phi_{,3}|_{\mathbf{Q}-\mathbf{p}} &= \frac{1}{2\Delta x} (\phi_{\mathbf{p}+[0,0,1]} - \phi_{\mathbf{p}+[0,0,-1]}) \end{aligned} \quad (33)$$

Consequently, Eq. 31 has the property that only  $\nabla^2\phi|_{\mathbf{Q}}$  contains a contribution from the central node,  $\phi_{\mathbf{p}}$  and thus

$$\frac{\partial}{\partial\phi_{\mathbf{p}}}(\phi_{,ij}g_{ij})|_{\mathbf{Q}} = \frac{1}{3}\text{tr}(\mathbf{g})|_{\mathbf{Q}-\mathbf{p}}\frac{\partial}{\partial\phi_{\mathbf{p}}}\nabla^2\phi|_{\mathbf{Q}} = -\frac{128}{90}\text{tr}(\mathbf{g})|_{\mathbf{Q}-\mathbf{p}}. \quad (34)$$

This is important for the Jacobi linearisation described in the next section. The PDE for  $\phi$  is thus approximated by ODEs at each point,  $\mathbf{p}$ , by

$$\dot{\phi}_{\mathbf{p}} = F_{\mathbf{p}}^{\phi}(\phi_{\mathbf{Q}}, U_{\mathbf{p}}, \theta_{\mathbf{p}}) \quad (35)$$

where

$$F_{\mathbf{p}}^{\phi} \equiv \frac{\frac{1}{3}\nabla^2\phi|_{\mathbf{Q}}\text{tr}(\mathbf{g})|_{\mathbf{Q}-\mathbf{p}} + M_{ij}|_{\mathbf{Q}-\mathbf{p}}g_{ij}|_{\mathbf{Q}-\mathbf{p}} - \frac{\partial f}{\partial\phi}(\phi_{\mathbf{p}}, U_{\mathbf{p}}, \theta_{\mathbf{p}})}{\tau(c_{\mathbf{p}}, \phi_{\mathbf{p}})A^2|_{\mathbf{Q}-\mathbf{p}}}. \quad (36)$$

In the above  $\tau(c_{\mathbf{p}}, \phi_{\mathbf{p}})$  is given by Eq. 8 and  $\frac{\partial f}{\partial\phi}|_{\mathbf{p}}$  is given by Eq. 15 using the values for  $\phi, U, \theta$  at point  $\mathbf{p}$ . The Laplacian  $\nabla^2\phi|_{\mathbf{Q}}$  is given by Eq. 27 with weights Eq. 30. The indexed functions  $M_{ij}$  are given by Eq. 32 and the functions  $g_{ij}$  are given by Eq. 23, where  $A, X_i, |\nabla\phi|^2$  are all functions of  $\phi_{,i}$  approximated by second order differences, Eq. 33. The PDE for  $U$ , Eq. 11, is approximated by the ODEs

$$\dot{U}_{\mathbf{p}} = F_{\mathbf{p}}^U(\phi_{\mathbf{p}}, \phi_{\mathbf{p}}, U_{\mathbf{Q}}, \theta_{\mathbf{p}}) \quad (37)$$

where

$$F_{\mathbf{p}}^U \equiv \frac{\nabla \cdot \left\{ D_c \frac{1-\phi_{\mathbf{p}}}{2} \nabla U|_{\mathbf{Q}} + \mathbf{j} \right\} + \frac{1}{2}[1 + (1 - k_E)U_{\mathbf{p}}]\dot{\phi}_{\mathbf{p}}}{\left( \frac{1+k_E}{2} - \frac{1-k_E}{2}\phi_{\mathbf{p}} \right)} \quad (38)$$

which is expanded in full (shown in subsection 3.3) with the same derivative discretisation scheme used for  $U$  as  $\phi$ .

Finally the  $\theta$  term is given by

$$\dot{\theta}_{\mathbf{p}} = F_{\mathbf{p}}^{\theta}(\theta_{\mathbf{Q}}, \dot{\phi}_{\mathbf{p}}) \equiv D_{\theta}\nabla^2\theta|_{\mathbf{Q}} + \frac{1}{2}\dot{\phi}_{\mathbf{p}}. \quad (39)$$

### 3.2 Boundary and initial conditions

We use zero Neumann boundary conditions for all variables. This is easily implemented by imposing values to the ghost cells of all blocks adjacent to the Neumann boundary, that are equal to the cell values of those cells next to the boundary (see below for more discussion of ghost cells). In the discretisation, this sets all boundary derivatives equal to zero. In exploitation of the symmetry in the problem this is interpreted as reflective symmetry on the planes  $x = 0, y = 0, z = 0$  and, provided the domain is sufficiently large, as equivalent to Dirichlet conditions on the far boundary for all variables. If, during a simulation, the normal derivative of any of the dependent variables begins to deviate from zero by more than a prescribed tolerance then we may allow the domain to expand so as to ensure we retain a zero normal derivative on the revised boundary (see below for more details of the mesh adaptivity that facilitates this). For a fixed domain, it is important that the dimensions are sufficient not only to represent the growing dendrite but also the temperature field throughout the simulation, which typically extends well ahead of the phase interface (especially for large values of the Lewis number).

The initial conditions for this problem are to some extent flexible as the evolution of the variables in time will alter physically inappropriate starting conditions. Thus the initial phase profile, temperature and concentration field, in general, will all adjust in the very early stages of the simulation. The temperature field can take longer to adjust to a profile which is near the melting point of the alloy inside the solid if started at a constant field value and, so we anticipate this with the condition given below.

The initial condition for the phase field with seed radius given by  $R$  is prescribed by

$$\phi(t = 0, \mathbf{x}(\mathbf{p})) = -\tanh[\alpha(\sqrt{\mathbf{x} \cdot \mathbf{x}} - R)], \quad (40)$$

where we employ the factor  $\alpha = 0.6$ , the precise value of which is not important as the solver smooths the phase profile if  $\alpha$  is large and conversely sharpen the profile if  $\alpha$  is too small within reason. It is not found necessary to normalise this profile so that  $\phi(t = 0, \mathbf{x} = \mathbf{0}) = 1$ . The initial solute condition is  $U = 0$  and the temperature profile used is

$$\theta(t = 0, \mathbf{x}(\mathbf{p})) = -\Delta + \frac{1}{2}\Delta(\phi + 1). \quad (41)$$

The most significant parameter in the initial conditions, in terms of the sensitivity of the subsequent calculations, is the radius of the initial nucleus. It has been shown that the transient behaviour of the evolving dendrite can be affected by this value well into the simulation, [11] (though the final geometry and velocity of the dendrite tip is much less sensitive). To this end we choose the smallest value of  $R$  such that the dendrite does not melt (melting can occur if there is insufficient solid  $\phi = 1$  in the nucleus due the encroachment of the diffuse interface near the nucleus centre). We find the smallest value to be  $R \approx 5$ .

### 3.3 Temporal discretisation

Due to the stiffness of the nonlinear system of ODEs that arises following the spatial discretization we employ BDF2 time stepping, so that at a point,  $\mathbf{p}$  in the grid domain at the centre the  $3^3$  points,  $\mathbf{Q}$ , the phase field variable system is approximated by

$$\phi_{\mathbf{p}}^{n+1} - r_2\phi_{\mathbf{p}}^n + r_3\phi_{\mathbf{p}}^{n-1} = r_1\Delta t^{n+1}F_{\mathbf{p}}^{\phi} \quad (42)$$

In practice, we introduce another variable

$$\phi_{\mathbf{p}}^* \equiv r_2\phi_{\mathbf{p}}^n - r_3\phi_{\mathbf{p}}^{n-1} \quad (43)$$

and write Eq. 42 as

$$\phi_{\mathbf{p}}^{n+1} - \phi_{\mathbf{p}}^* = r_1\Delta t^{n+1}F_{\mathbf{p}}^{\phi}(\phi_{\mathbf{Q}}^{n+1}, U_{\mathbf{p}}^{n+1}, \theta_{\mathbf{p}}^{n+1}). \quad (44)$$

The right hand side,  $F_{\mathbf{p}}^{\phi}$ , is defined by Eq. 36.

For constant time step,  $r_1 = 2/3, r_2 = 4/3, r_3 = 1/3$ . For a growing dendrite it is essential to use a small time step at the initial (imposed) state. Thereafter the time step is increased (see subsection 4.2 for detail) to fully exploit the implicit time stepping. The BDF2 for adaptive time stepping is given by

$$\begin{aligned} r_1 &\equiv (r + 1)/(2r + 1), \\ r_2 &\equiv (r + 1)^2/(2r + 1), \\ r_3 &\equiv r^2/(2r + 1) \\ r &\equiv \Delta t^{n+1}/\Delta t^n \end{aligned} \quad (45)$$



where  $r$  is the ratio of the current to previous time step.

Similarly, with  $U_{\mathbf{p}}^* \equiv r_2 U_{\mathbf{p}}^n - r_3 U_{\mathbf{p}}^{n-1}$ , we write

$$U_{\mathbf{p}}^{n+1} - U_{\mathbf{p}}^* = r_1 \Delta t^{n+1} F_{\mathbf{p}}^U(\phi_{\mathbf{Q}}^{n+1}, \phi_{\mathbf{p}}^*, U_{\mathbf{Q}}^{n+1}, U_{\mathbf{p}}^*, \theta_{\mathbf{p}}^{n+1}) \quad (46)$$

where

$$\begin{aligned} F_{\mathbf{p}}^U(\phi_{\mathbf{Q}}, \phi_{\mathbf{p}}^*, U_{\mathbf{Q}}, U_{\mathbf{p}}^*, \theta_{\mathbf{p}}) &\equiv \left\{ \frac{\nabla \cdot \left( D_c \frac{1-\phi}{2} \nabla U + \mathbf{j} \right) + \frac{1}{2} [1 + (1 - k_E) U] \frac{\partial \phi}{\partial t}}{\left( \frac{1+k_E}{2} - \frac{1-k_E}{2} \phi \right)} \right\} \bigg|_{\mathbf{Q}} \\ &\equiv \frac{D_c \frac{1-\phi}{2} \nabla^2 U|_{\mathbf{Q}} + \nabla(D_c \frac{1-\phi}{2})|_{\mathbf{Q}-\mathbf{p}} \cdot \nabla U|_{\mathbf{Q}-\mathbf{p}} + \nabla \cdot \mathbf{j}|_{\mathbf{Q}} + \frac{1}{2} [1 + (1 - k_E) U_{\mathbf{p}}] \frac{\phi_{\mathbf{p}} - \phi_{\mathbf{p}}^*}{r_1 \Delta t^{n+1}}}{\left( \frac{1+k_E}{2} - \frac{1-k_E}{2} \phi_{\mathbf{p}} \right)}. \end{aligned} \quad (47)$$

The divergence of the anti-trapping current is given by

$$\nabla \cdot \mathbf{j}|_{\mathbf{Q}} = -\frac{1}{2\sqrt{2}} [1 + (1 - k_E)] \nabla \cdot (U \dot{\phi} \mathbf{n})|_{\mathbf{Q}} \quad (48)$$

where

$$\begin{aligned} \nabla \cdot (U \dot{\phi} \mathbf{n})|_{\mathbf{Q}} &= \dot{\phi} \nabla U \cdot \mathbf{n} + U \nabla \dot{\phi} \cdot \mathbf{n} + U \dot{\phi} \nabla \cdot \mathbf{n} \\ &= \frac{\phi_{\mathbf{p}}^{n+1} - \phi_{\mathbf{p}}^*}{r_1 \Delta t^{n+1}} \nabla U|_{\mathbf{Q}-\mathbf{p}} \cdot \mathbf{n}|_{\mathbf{Q}-\mathbf{p}} + U_{\mathbf{p}} \frac{\nabla \phi^{n+1}|_{\mathbf{Q}-\mathbf{p}} - \nabla \phi^*|_{\mathbf{Q}-\mathbf{p}}}{r_1 \Delta t^{n+1}} \cdot \mathbf{n}|_{\mathbf{Q}-\mathbf{p}} + U_{\mathbf{p}} \frac{\phi_{\mathbf{p}}^{n+1} - \phi_{\mathbf{p}}^*}{r_1 \Delta t^{n+1}} \nabla \cdot \mathbf{n}|_{\mathbf{Q}} \end{aligned} \quad (49)$$

and

$$\begin{aligned} \nabla \cdot \mathbf{n}|_{\mathbf{Q}} &\equiv \frac{\partial}{\partial x^a} \left( \frac{\phi_{,a}}{\sqrt{\phi_{,b} \phi_{,b}}} \right) \bigg|_{\mathbf{Q}} = \frac{1}{\sqrt{\nabla \phi \cdot \nabla \phi}|_{\mathbf{Q}-\mathbf{p}}} \left( \nabla^2 \phi|_{\mathbf{Q}} - \frac{\phi_{,a} \phi_{,ab} \phi_{,b}}{\nabla \phi \cdot \nabla \phi} \bigg|_{\mathbf{Q}} \right) \\ &\equiv \frac{1}{\sqrt{\nabla \phi \cdot \nabla \phi}|_{\mathbf{Q}-\mathbf{p}}} (\nabla^2 \phi|_{\mathbf{Q}} - n_a|_{\mathbf{Q}-\mathbf{p}} \phi_{,ab}|_{\mathbf{Q}} n_b|_{\mathbf{Q}-\mathbf{p}}). \end{aligned} \quad (50)$$

Finally, the heat equation with  $\theta_{\mathbf{p}}^* \equiv r_2 \theta_{\mathbf{p}}^n - r_3 \theta_{\mathbf{p}}^{n-1}$  we write

$$\theta_{\mathbf{p}}^{n+1} - \theta_{\mathbf{p}}^* = r_1 \Delta t^{n+1} F_{\mathbf{p}}^\theta(\phi_{\mathbf{p}}^{n+1}, \phi_{\mathbf{p}}^*, \theta_{\mathbf{Q}}^{n+1}, \theta_{\mathbf{p}}^*) \quad (51)$$

where

$$F_{\mathbf{p}}^\theta(\phi_{\mathbf{p}}, \phi_{\mathbf{p}}^*, \theta_{\mathbf{Q}}, \theta_{\mathbf{p}}^*) \equiv D_\theta \nabla^2 \theta|_{\mathbf{Q}} + \frac{1}{2} \frac{\phi_{\mathbf{p}} - \phi_{\mathbf{p}}^*}{r_1 \Delta t^{n+1}} \quad (52)$$

### 3.4 Adaptive mesh and block tree structure

The domain is first divided into a number of mesh blocks each of which contains  $N \times N \times N$  hexahedral cells, where we typically choose  $N = 8$ . We employ a domain of  $800^3$ , which is large enough for Lewis numbers of the order of 100. We divide this domain into  $4^3$  blocks, so that when  $N=8$ , each cell is of size  $25^3$  and refer to this as level 1. The adaptive mesh strategy then imposes a hierarchical sub-division of some or all of these blocks, and their descendants, based upon an oct tree structure. This subdivision aims to concentrate cells where gradients of the oct tree variables are highest and to ensure neighbour blocks differ by at most one level. The finest grid we work with is at level 7, with a corresponding  $\Delta x = 25/2^7 = 0.1953125$ . We find that the minimum finest level necessary to obtain qualitatively reasonable results is level 5, corresponding to  $\Delta x = 0.78125$ . As noted previously, throughout this work we exploit cell-centred finite differences in our discretization, in which a single unknown is associated with the centre of each hexahedral cell for each of the dependent variables.

In order to discuss further the tree structure of the blocks we denote any block by its label,  $i$  and its contents/properties,  $B_i$ :

$$B(i) = [l, s, p, \mathbf{c}, \mathbf{x}] \quad (53)$$

where  $l \in [1, n]$  is the level,  $s \in [1, 8]$  is the sibling number (i.e. an index for which child of  $p$  the block is),  $p$  is the parent index,  $c_i, i \in [1, 8]$  are the 8 child indices, and  $\mathbf{x} = [x, y, z]$  is the Cartesian position coordinates of the block origin. Any one of these properties can be accessed by the block number,  $i$ . Some examples:  $p(i)$  is the block number

of the parent of block  $i$ ;  $\mathbf{x}(p(i))$  is the position of the parent's origin;  $c_{s(i)}(p(i)) = i$  is an identity. A natural position scheme for the child blocks is

$$\begin{aligned}
\mathbf{x}(c_1(i)) &= \mathbf{x}(i) + \frac{1}{2}\Delta x[-1, -1, -1] \\
\mathbf{x}(c_2(i)) &= \mathbf{x}(i) + \frac{1}{2}\Delta x[1, -1, -1] \\
\mathbf{x}(c_3(i)) &= \mathbf{x}(i) + \frac{1}{2}\Delta x[-1, 1, -1] \\
\mathbf{x}(c_4(i)) &= \mathbf{x}(i) + \frac{1}{2}\Delta x[1, 1, -1] \\
\mathbf{x}(c_5(i)) &= \mathbf{x}(i) + \frac{1}{2}\Delta x[-1, -1, 1] \\
\mathbf{x}(c_6(i)) &= \mathbf{x}(i) + \frac{1}{2}\Delta x[1, -1, 1] \\
\mathbf{x}(c_7(i)) &= \mathbf{x}(i) + \frac{1}{2}\Delta x[-1, 1, 1] \\
\mathbf{x}(c_8(i)) &= \mathbf{x}(i) + \frac{1}{2}\Delta x[1, 1, 1]
\end{aligned} \tag{54}$$

where  $\Delta x$  is the grid size associated with level  $i$ . A complete specification of all the blocks in the oct tree is then specified by the list:

$$\mathbf{B} = \{B(i), i \in [1, B_N]\} \tag{55}$$

where  $B_N$  is the total block number. Moreover, a childless block,  $i$ , can be indicated by specifying,  $\mathbf{c}(i) = 0$  and so the oct tree also can be specified by a listing of just the leaf blocks

$$\mathbf{B} = \{B(i), i \in [1, B_N] : \mathbf{c}(i) = \mathbf{0}\}. \tag{56}$$

There is no adaptive meshing within each block, which always contains  $N \times N \times N$  cells and the adaptive strategy is further restricted by only allowing blocks at level  $n$  adjacent to blocks of  $n-1$ ,  $n$  and  $n+1$ . Thus, even though a block may be flagged for coarsening, this (latter) restriction often prevents this happening. Conversely, blocks flagged for refinement must, if necessary, be accompanied by refinement on neighbouring blocks. Blocks are flagged for refinement if, for any point,  $\mathbf{p}$ , in the block, the following criterion is satisfied:

$$e \equiv \max \{e_\phi |\phi_{\mathbf{p}} - \phi_{\mathbf{p}-\mathbf{q}}|, e_U |U_{\mathbf{p}} - U_{\mathbf{p}-\mathbf{q}}|, e_T |T_{\mathbf{p}} - T_{\mathbf{p}-\mathbf{q}}|\} > \eta, \tag{57}$$

where we use, for tolerance,  $\eta \sim 1$  and

$$|\phi_{\mathbf{p}} - \phi_{\mathbf{p}-\mathbf{q}}| \equiv \sqrt{(\phi_{\mathbf{p}} - \phi_{\mathbf{p}-[1,0,0]})^2 + (\phi_{\mathbf{p}} - \phi_{\mathbf{p}-[0,1,0]})^2 + (\phi_{\mathbf{p}} - \phi_{\mathbf{p}-[0,0,1]})^2}, \tag{58}$$

etc. Typically the weights,  $e_\phi$ ,  $e_U$  and  $e_T$  are chosen to sum to unity. Sometimes it is convenient to set  $e_U$  to zero to suppress unnecessary refinement within the solid. If  $e < 0.1\eta$ , the block is flagged for derefinement.

Although each block is logically defined to be of dimension  $N \times N \times N$  the PARAMESH implementation actually allocates a block of dimension  $(N + 2G) \times (N + 2G) \times (N + 2G)$ , where  $G$  is the number of guard cells (sometimes referred to as ghost cells) around each block. When  $G = 1$ , as used in this paper, the first and last cells in each dimension are guard cells - an update function may be called at any time in order to populate these guard cells with the corresponding values from the interior of each of the neighbouring blocks (with a separate treatment required to impose boundary conditions for blocks at the edge of the domain, as discussed previously). The application of a discrete stencil on any block requires access to neighbouring blocks via the guard cell nodes. When the neighbouring block is coarser the guard cell of the coarse block is found by a weighted average of the 8 surrounding coarse nodes (some of which are in the parent block). Using a tri-linear function of  $x, y$  and  $z$ , gives the weightings, in order of nearest neighbours first

$$\mathbf{w} = \left[ \frac{27}{64}, \frac{9}{64}, \frac{9}{64}, \frac{9}{64}, \frac{3}{64}, \frac{3}{64}, \frac{3}{64}, \frac{1}{64} \right]. \tag{59}$$

The process is known as prolongation. For example, the value of  $\phi$  at a fine node is prolonged by

$$\phi_{\text{fine}} = \sum_{i=\text{coarse cube centres}} w_i \phi_i. \tag{60}$$

The reverse process, of finding a guard cell value for a coarse block when one or more neighbours is refined is known as restriction and is the simple average of the eight nearest, one level finer, cell centres. Both operations, restriction and prolongation using Eq. 59, are also required for multigrid as detailed in the next section.

## 4 Solver method

The discretisation above produces a system of nonlinear algebraic equations for  $\phi_{\mathbf{p}}^{n+1}$ ,  $U_{\mathbf{p}}^{n+1}$  and  $\theta_{\mathbf{p}}^{n+1}$  at each time step,  $t^{n+1}$ . In this section we describe the solution algorithm that is used to solve these systems, based upon a nonlinear multigrid (Full Approximation Scheme (FAS), [12]) approach. Initially we describe the nonlinear smoother upon which the multigrid is built, before exploring how the multigrid solver combines this with the hierarchical mesh adaptivity introduced in the previous section. Finally, in subsection 4.3, we explain the key features of the parallel implementation, including the issues associated with parallel dynamic load-balancing.

### 4.1 Nonlinear smoother

The non-linear system of algebraic equations at the end of Sec. 3.3 can be written using the generic vector notation  $\mathbf{v}_{\mathbf{p}}^{n+1} \equiv [\phi_{\mathbf{p}}^{n+1}, \theta_{\mathbf{p}}^{n+1}, U_{\mathbf{p}}^{n+1}]$  by

$$\mathbf{A}_{\mathbf{p}}(\mathbf{v}_{\mathbf{Q}}^{n+1}) = \mathbf{0} \quad (61)$$

where

$$\mathbf{A}_{\mathbf{p}}(\mathbf{v}_{\mathbf{Q}}^{n+1}) \equiv \mathbf{v}_{\mathbf{p}}^* - \mathbf{v}_{\mathbf{p}}^{n+1} + r_1 \Delta t^{n+1} \mathbf{F}_{\mathbf{p}}(\mathbf{v}_{\mathbf{Q}}^{n+1}) \quad (62)$$

for each node,  $\mathbf{p}$ , in the grid. Recall, that the appearance of  $\mathbf{Q}$  indicates coupling between points  $\mathbf{p}$  and neighbouring nodes, and the BDF2 notation  $\mathbf{v}_{\mathbf{p}}^*$  in combination with  $r_1$  are defined in subsection 3.3. Using an iteration method, with  $\mathbf{v}_{\mathbf{p}}^{n+1}$  approximated by  $\mathbf{v}_{\mathbf{p}}^{n+1,m}$ , we define the defect

$$\mathbf{d}_{\mathbf{p}}^{n+1,m} = -\mathbf{A}_{\mathbf{p}}(\mathbf{v}_{\mathbf{Q}}^{n+1,m}). \quad (63)$$

The pointwise Newton update for this iteration is given by

$$\mathbf{v}_{\mathbf{p}}^{n+1,m+1} = \mathbf{v}_{\mathbf{p}}^{n+1,m} - \omega \tilde{\mathbf{d}}_{\mathbf{p}}^{n+1,m}, \quad (64)$$

where  $\tilde{\mathbf{d}}_{\mathbf{p}}^{n+1,m}$  is found by solving the  $3 \times 3$  system

$$\mathbf{J}_{\mathbf{p}}^{n+1,m} \tilde{\mathbf{d}}_{\mathbf{p}}^{n+1,m} \equiv \mathbf{d}_{\mathbf{p}}^{n+1,m} \quad (65)$$

with the  $3 \times 3$  Jacobian matrix defined by

$$\mathbf{J}_{\mathbf{p}}^{n+1,m} \equiv \frac{\partial \mathbf{d}_{\mathbf{p}}^{n+1,m}}{\partial \mathbf{v}_{\mathbf{p}}^{n+1,m}}. \quad (66)$$

In practice we typically select,  $\omega \approx 0.9$  and find that off diagonal terms of  $\mathbf{J}_{\mathbf{p}}^{n+1,m}$  are not essential to obtain a convergent iteration.

The precise form of the entries of  $\mathbf{J}_{\mathbf{p}}^{n+1,m}$  may be deduced from the above. However, we illustrate this in detail for one of the diagonal terms for the sake of clarity. Denoting the diagonal entries of  $\mathbf{J}_{\mathbf{p}}^{n+1,m}$  by  $[J_{\mathbf{p}}^{\phi,n+1,m}, J_{\mathbf{p}}^{\theta,n+1,m}, J_{\mathbf{p}}^{U,n+1,m}]$ , using Eqs. 44 and 36, and treating terms not including  $\phi_{\mathbf{p}}$  as constant we find

$$J_{\mathbf{p}}^{\phi,n+1,m} \equiv \frac{\partial d_{\mathbf{p}}^{\phi}(\mathbf{v}_{\mathbf{Q}}^{n+1,m})}{\partial \phi_{\mathbf{p}}} = \left[ 1 + r_1 \Delta t^{n+1} \frac{1}{3} \frac{g_{11} + g_{22} + g_{33}}{\tau(c, \phi) A^2} \frac{128}{30(\Delta x)^2} \right]_{\mathbf{p}}, \quad (67)$$

where we note that the contribution from the central node to  $\nabla^2 \phi$  is

$$\frac{\partial \nabla^2 \phi|_{\mathbf{Q}}}{\partial \phi_{\mathbf{p}}} = -\frac{128}{30(\Delta x)^2}. \quad (68)$$

The new updated solution for  $\phi$  at time  $t^{n+1}$  at iteration  $m$  and point  $\mathbf{p}$  is given by

$$\phi_{\mathbf{p}}^{n+1,m+1} = \phi_{\mathbf{p}}^{n+1,m} - \omega \frac{d_{\mathbf{p}}^{\phi,n+1,m}}{J_{\mathbf{p}}^{\phi,n+1,m}}. \quad (69)$$

The term  $J_{\mathbf{p}}^{\theta,n+1,m} = 1 + r_1 \Delta t^{n+1} D_{\theta} \frac{128}{30(\Delta x)^2}$  and, though the term  $J_{\mathbf{p}}^{U,n+1,m}$  has many components, the linearity of  $F_{\mathbf{p}}^U$  in  $U_{\mathbf{p}}$  leads to straightforward updates for the  $U$  components, n.b. gradients involving  $U|_{\mathbf{Q}-\mathbf{p}}$  are treated as constant.

### FAS algorithm to solve $\mathbf{A}(\mathbf{v}) = \mathbf{0}$

1.  $h = \text{finest grid (top level)}$
2.  $\mathbf{v}_{\mathbf{p}(h)}^{n+1,0} \leftarrow \mathbf{v}_{\mathbf{p}(h)}^n$ : set initial guess equal to value at last time step
3.  $\mathbf{v}_{\mathbf{p}(h)}^{n+1,\nu+1} \leftarrow \mathbf{V}\text{-cycle}(\mathbf{v}_{\mathbf{p}(h)}^{n+1,\nu}, \mathbf{0}, h)$ : application of V-cycle,  $\nu = 0, 1, \dots$  until convergence

### Recursive Function $\mathbf{V}\text{-cycle}(\mathbf{v}_{\mathbf{p}(h)}^0, \mathbf{f}_{\mathbf{p}(h)}, h) \rightarrow \mathbf{v}_{\mathbf{p}(h)}^{m+1}$ (solves $\mathbf{A}(\mathbf{v}) = \mathbf{f}$ )

1.  $\mathbf{v}_{\mathbf{p}(h)}^{m+1} \leftarrow \mathbf{S}(\mathbf{v}_{\mathbf{p}(h)}^m, \mathbf{f}_{\mathbf{p}(h)}, h) \equiv \begin{cases} \mathbf{d}_{\mathbf{p}(h)}^m \leftarrow \mathbf{f}_{\mathbf{p}(h)} - \mathbf{A}(\mathbf{v}_{\mathbf{Q}(h)}^m) \\ \mathbf{v}_{\mathbf{p}(h)}^{m+1} \leftarrow \mathbf{v}_{\mathbf{p}(h)}^m - \omega \left[ \frac{d_{\mathbf{p}(h)}^{\phi,m}}{J_{\mathbf{p}(h)}^{\phi,m}}, \frac{d_{\mathbf{p}(h)}^{\theta,m}}{J_{\mathbf{p}(h)}^{\theta,m}}, \frac{d_{\mathbf{p}(h)}^{U,m}}{J_{\mathbf{p}(h)}^{U,m}} \right] \end{cases}, \quad m = 0, \dots, M-1$   
(pre) smooth  $M$  times, (typically we use  $M \sim 4$ )
2.  $\mathbf{d}_{\mathbf{p}(h)}^{m+1} \leftarrow \mathbf{f}_{\mathbf{p}(h)} - \mathbf{A}_{\mathbf{p}}(\mathbf{v}_{\mathbf{Q}(h)}^{m+1})$   
 $\mathbf{v}_{\mathbf{p}(2h)}^0 \leftarrow \mathbf{I}_{2h}^h(\mathbf{v}_{\mathbf{p}(h)}^{m+1})$   
 $\mathbf{d}_{\mathbf{p}(2h)}^0 \leftarrow \mathbf{I}_{2h}^h(\mathbf{d}_{\mathbf{p}(h)}^{m+1}) + \mathbf{A}(\mathbf{v}_{\mathbf{p}(2h)}^0)$ : restriction
3. **if** not coarsest level  
     $\mathbf{v}_{\mathbf{p}(2h)}^{m+1} \leftarrow \mathbf{V}\text{-cycle}(\mathbf{v}_{\mathbf{p}(2h)}^0, \mathbf{d}_{\mathbf{p}(2h)}^0, 2h)$ .  
**else** (bottom level)  
     $\mathbf{v}_{\mathbf{p}(2h)}^{m+1} \leftarrow \mathbf{S}(\mathbf{v}_{\mathbf{p}(2h)}^m, \mathbf{d}_{\mathbf{p}(2h)}^0, 2h), \quad m = 0, \dots, N$  (typically we use  $N \sim 4$ )  
**end if**
4.  $\mathbf{d}_{\mathbf{p}(2h)}^0 \leftarrow \mathbf{v}_{\mathbf{p}(2h)}^{m+1} - \mathbf{v}_{\mathbf{p}(2h)}^0$ : compute the correction (now on upward part of v-cycle)
5.  $\mathbf{v}_{\mathbf{p}(h)}^0 \leftarrow \mathbf{v}_{\mathbf{p}(h)}^{m+1} + \mathbf{I}_h^{2h}(\mathbf{d}_{\mathbf{p}(2h)}^0)$ : prolong and correct fine grid solution
6.  $\mathbf{v}_{\mathbf{p}(h)}^{m+1} \leftarrow \mathbf{S}(\mathbf{v}_{\mathbf{p}(h)}^m, \mathbf{0}, h), \quad m = 0, \dots, M_{\text{post}} - 1$ : (post) smooth with this new value  $M_{\text{post}} \sim 4$  times

Figure 2: The FAS algorithm

The above describes a point wise non-linear Jacobi smoother. The Jacobi approach lends itself to parallel implementation since the computation at each point may be completed using neighbouring values from the previous iteration only. Consequently, only one ghost cell update is required prior to each sweep through the mesh. This keeps inter processor communication to a minimum. Using the stencils described above it is possible to complete the updates using just one layer of ghost cells. Hence if a block size of  $8 \times 8 \times 8$  (say) is used in PARAMESH then a  $10 \times 10 \times 10$  block is actually allocated to accommodate the ghost layer of each block edge.

Having derived the point-wise Jacobi smoother, in the following subsection we show how, this may be used as part of a non-linear geometric multigrid solver combined with local mesh adaptivity. Discussion of the parallel implementation is postponed until subsection 4.3.

## 4.2 Nonlinear Multigrid

Although the Jacobi smoother described above gives a convergent iteration for the system Eq. 61 (for sufficiently small  $\Delta t$  and good initial guess), the convergence is far too slow to be of any practical use. Fortunately, however, the iteration also satisfies a smoothing property which means that it damps out the highest frequency components of the error (defect) far more quickly than the rest. This makes it ideally suited for use as part of a nonlinear multigrid scheme. In this work we make use of the Full Approximation Scheme (FAS) of Brandt [12]. We denote the value of variables at point,  $\mathbf{p}$ , time  $t^{n+1}$ , iteration,  $m$ , and level/grid size,  $h$  by  $\mathbf{v}_{\mathbf{p}(h)}^{n+1,m}$  and coarser level  $\mathbf{v}_{\mathbf{p}(2h)}^{n+1,m}$ . The restriction operation,  $\mathbf{I}_{2h}^h(\mathbf{v}_{\mathbf{p}(h)})$ , is the assignment to  $\mathbf{v}_{\mathbf{p}(2h)}$  of the simple average of the 8 surrounding nodes,  $\mathbf{v}_{\mathbf{p}(h)}$ . Prolongation,  $\mathbf{I}_h^{2h}\mathbf{v}_{\mathbf{p}(2h)}$ , is an assignment to  $\mathbf{v}_{\mathbf{p}(h)}$  given by applying Eq. 59 to give a weighted average of the values at the 8 nearest coarse nodes  $\mathbf{v}_{\mathbf{p}(2h)}$ . In solving Eq. 61, FAS computes a defect from the restricted defect and variables to give a modified  $\mathbf{A}(\mathbf{v}) = \mathbf{f}$  on these lower levels. See Fig. 2, where we detail FAS for our notation.

Note that in our work local mesh adaptivity is an essential feature. This has been described in Sec. 3.4, where PARAMESH block data structures are used as nodes of an oct-tree. In this work we also use the oct-tree as part of the geometric multigrid solver by developing an implementation of the Multi Level Adaptive Technique (MLAT) (see

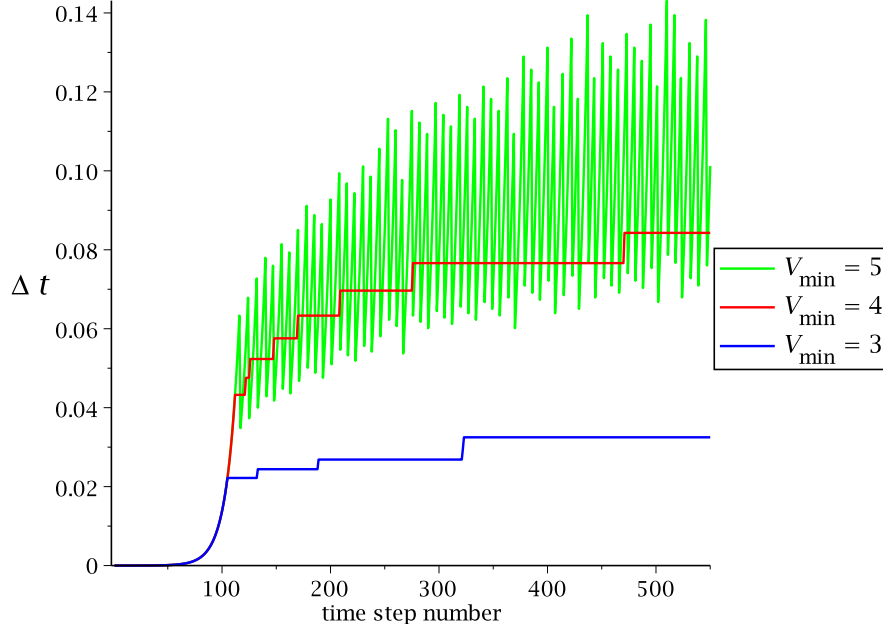


Figure 3: An illustration, for  $Le = 40$ ,  $\Delta = 0.525$  of the evolution of the time step,  $\Delta t$ , from  $10^{-6}$  by V-cycle control.

[12]). MLAT allows us to use the Jacobi smoother on each block without modification, provided the prolongation and restriction operators are adapted to deal with guard cells of the interface between too differing levels of refinement. Briefly, the smoother is only applied in the regions of the domain that contains fine level blocks, but the coarse grid correction takes place on all parts of the domain that contains the coarse level blocks (though the modified right hand side associated with the FAS scheme only contributes to those coarse grid regions that have fine grids on them).

For the results presented in the following sections we are primarily interested in obtaining solutions at large times. Hence we choose a time-stepping strategy with this in mind, based upon the number of nonlinear V-cycles that are required to achieve convergence (for an alternative strategy, which uses a local error estimate to control  $\Delta t$ , see [13]). The principle is simple: if the nonlinear multigrid converges easily at a give time step then increase  $\Delta t$ , whereas if it converges slowly (or fails to converge) then decrease  $\Delta t$  (repeating the time step in the case of failure). Convergence is deemed to have occurred when the infinity norm of the defect (possibly weighted differently for each dependent variable),  $d$ , satisfies  $d < d_{\max}$ , for a user-defined stopping parameter  $d_{\max}$ . If this is not satisfied in  $V_{\text{fail}}$  V-cycles then  $\Delta t$  is halved and the step is retaken. If convergence occurs in  $V_{\min}$  V-cycles or less then  $\Delta t$  is increased by 10% however if convergence occurs in more than  $V_{\max}$  V-cycles then  $\Delta t$  is halved for the next step. Figure 3 shows a typical evolution of the time step size for three different choices of  $V_{\min}$ , based upon initial  $\Delta t = 10^{-6}$ ,  $d_{\max} = 10^{-10}$  and  $V_{\max} = 10$ . Note that although the oscillation in  $\Delta t$  is aesthetically undesirable it has no adverse effect on the solution quality nor (so long as  $V_{\max} < V_{\text{fail}}$ ) the overall efficiency.

### 4.3 Parallel implementation

Our implementation requires communication between blocks both on the same level (to apply the smoothing steps) and between levels (for prolongation and restriction). For parallel processing, clearly, communication between cores needs to be kept to a minimum, but an important secondary consideration is that each core has as near as possible equal load. For a uniform mesh an allocation of each core to an equal volume of the computational domain results in an obvious fair division of labour. On the other hand, for an adaptive mesh, such an approach fails since the loading between cores will differ enormously.

Given the label,  $i$ , of each block in Eq. 53, we present the Morton ordering,  $M(i) \in [1, B_N]$ , of an adaptive mesh

$$M(i) = \begin{cases} 1 & l = 1, s = 1 \text{ (bottom level)} \\ M(p(i)) + 1, & s = 1, l > 1 \text{ (one plus parent's label)} \\ M(s(i-1)) + 1, & s > 2, \text{ and } l = \text{leaf level} \\ M(c_s(i)) + 1, & \text{otherwise, i.e. } s = 2 \end{cases}, \quad (70)$$

where ‘leaf level’ refers an unrefined level (parent without child) and includes the finest level. The blocks may then

Parameters		Tip radius		
$Le$	$\Delta$	$\Delta x = 0.78$	$\Delta x = 0.39$	$\Delta x = 0.195$
40	0.325	$39.0 \pm 2$	$39.7 \pm 0.3$	$39.7 \pm 0.1$
40	0.525	$29.7 \pm 2$	$31.5 \pm 0.3$	$31.8 \pm 0.1$
100	0.325	$40.3 \pm 2$	$39.8 \pm 0.3$	$39.8 \pm 0.1$

Table 2: Table of tip radius results for two under-coolings and two Lewis numbers. For this particular selection of parameters there is reasonable agreement even on the coarser mesh  $\Delta x = 0.78$ . But the slight discrepancy shown for the higher under-cooling is symptomatic of the necessity for a finer mesh,  $\Delta x > 0.39$ , in general.

be put into a Morton ordered list:

$$\begin{aligned} \mathbf{M}(\mathbf{B}) &\equiv \{B_i : j = M(i), j \in [1, B_N]\} \\ &\equiv \{\dots, B_{M(i)}, B_{M(j)}, \dots : M(i) < M(j)\} \end{aligned} \quad (71)$$

We write the form given in second line of Eq. 71 to highlight the difference between Morton ordering and the alternative ordering we adopt below, Eq. 72 and 73.

Load balancing for  $B_N$  blocks on processors  $p_i, i = 1, 2, \dots, c_N$  follows the same order with approximately  $B_N/c_N$  blocks per core. The resulting ordering is well known to exhibit parallel inefficiency for non-uniform meshes. This is because Morton ordering on adaptive meshes leaves the majority of the top level blocks on a small fraction of cores and since multigrid advances from the top to bottom level and back sequentially, the majority of the cores will be idle at the top level. In three dimensions this problem is acute because there is a factor of 8 between levels.

We adopt the following ordering, which may be termed Morton-*Level* ordering.

$$\begin{aligned} \mathbf{L}(\mathbf{B}) &\equiv \{B_i : j = L(i), j \in [1, B_N]\}, \\ &\equiv \{\mathbf{L}_k(\mathbf{B}), k \in [1, n]\}, \end{aligned} \quad (72)$$

where the subsets,  $\mathbf{L}_k(\mathbf{B})$  on each level,  $k$  are defined

$$\mathbf{L}_k(\mathbf{B}) \equiv \{\dots, B_{M(i)}, B_{M(j)}, \dots : M(i) < M(j), l(i) = l(j) = k\} \quad (73)$$

In summary, we implement a depth first traversal of the blocks and then, using this numbering, divide the work load at each level in turn between all processors. For a uniform mesh this strategy results in a near optimum allocation to cores. For adaptive meshes the communication on any level is also optimum, but communication between level is compromised.

## 5 Computational Results

This section provides a selection of computational results that are designed to validate and assess our proposed solution algorithm. Since this is the first attempt to produce three dimensional results for the solidification of a non-isothermal alloy using a realistic interface width we have no external simulations against which to validate our code. Hence the approach taken here has been to firstly validate a two-dimensional restriction of our implementation against our own two-dimensional solver, implemented completely independently and described in [13]. These tests show that we are indeed able to reproduce results from [2], (e.g. the tip radius and velocities are in excellent agreement) even though the two code bases are completely independent, e.g. in [2] forth order accurate stencils were used.

Furthermore, we have also successfully validated two 3-d simplifications of our implementation. In [14] we consider a thermal-only restriction (i.e. a pure metal, so no concentration equation present), where we show quantitative agreement with results obtained using the 3-d, explicit, thermal-only approach of [15, 16]. Similarly, in [17] we show quantitative agreement between an isothermal version of our solver (i.e. no temperature equation present) and another 3-d isothermal solidification code described in [15]. To validate the 3-D non-isothermal simulations we now rely on having mesh convergence and multigrid performance.

The remainder of this section is divided into two subsections. The first of these considers the mesh convergence of our implementation, showing that large time solutions obtained on a sequence of finer levels of maximum refinement do indeed appear to converge to particular dendrite geometries, as tested for a selection of parameter values. The second subsection considers the numerical performance of the solver. In particular it is shown that optimal performance is achieved, whereby the time required to complete a time step grows almost linearly with the total number of degrees of freedom. The capability improvements associated with the distributed memory parallel implementation are also discussed.

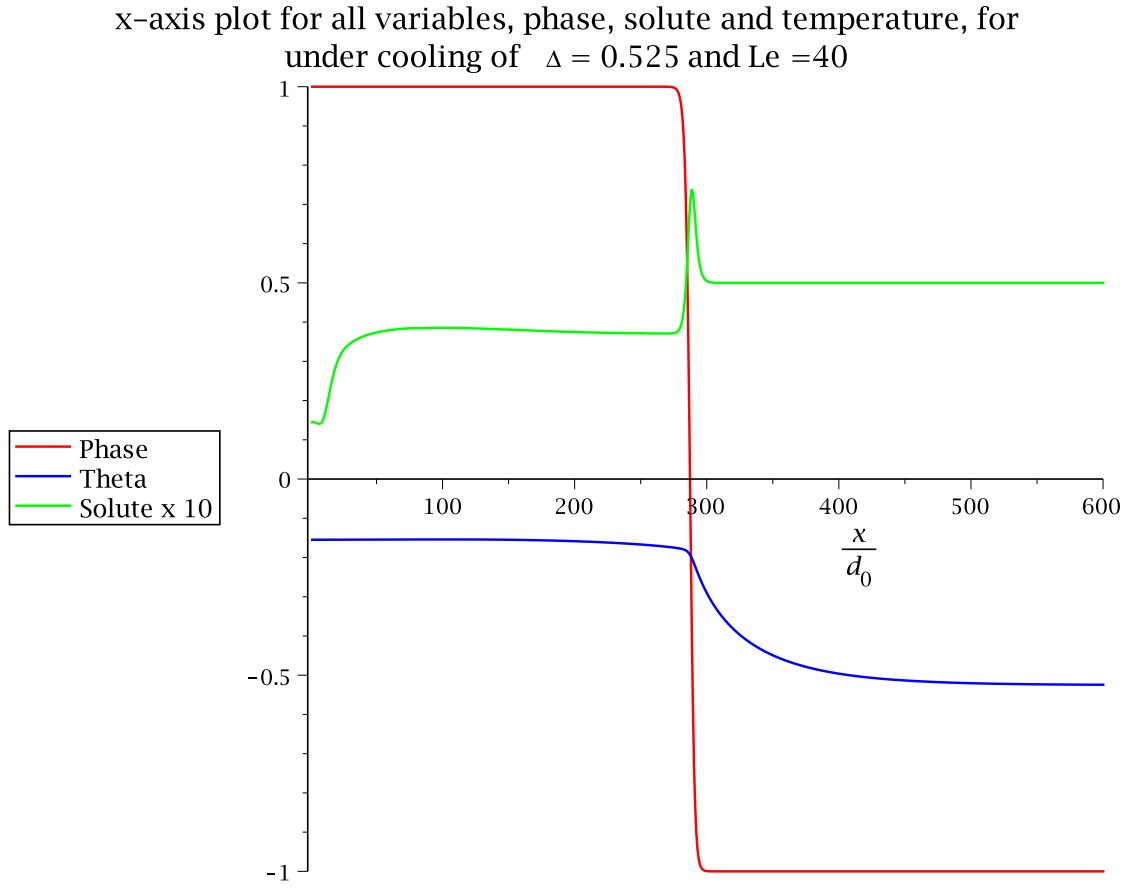


Figure 4: This plot shows the variables: phase  $\phi$ , solute concentration  $c$ , and dimensionless temperature  $\theta$ . At Lewis number of 40 and under cooling of 0.525 the temperature diffusion zone extends to about 300 in a domain size of  $800^3$

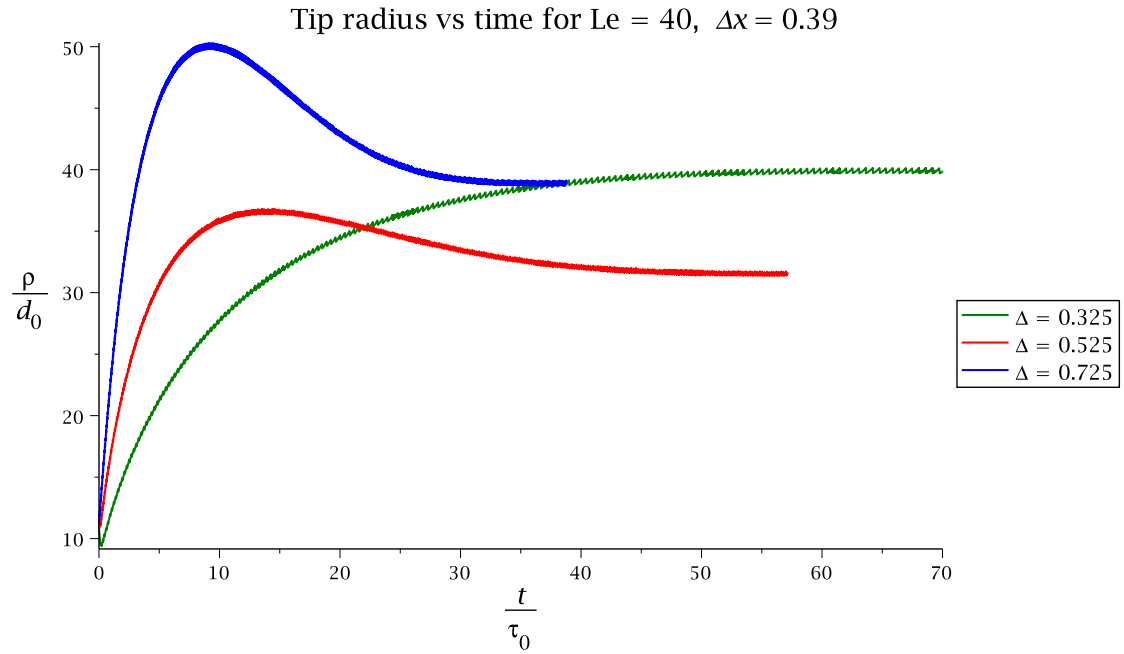


Figure 5: Tip radius at  $dx = 0.39$  and  $Le = 40$  for a range of under cooling  $\Delta$ . Even though the radius at  $\Delta = 0.325$  takes longer to reach steady state, this simulation is faster than the others due to greater stability resulting in fewer, larger time steps.

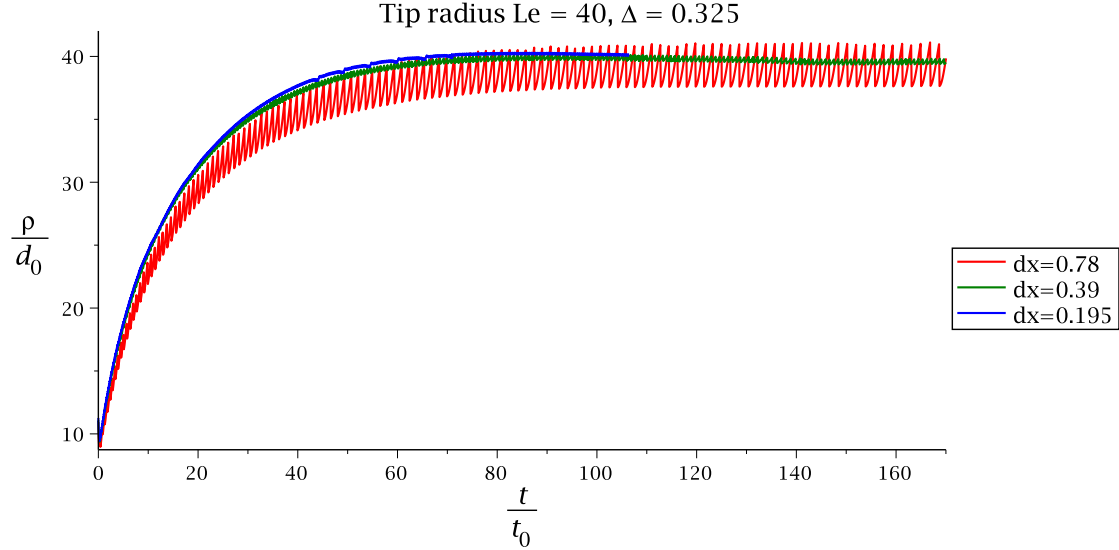


Figure 6: Convergence test on tip radius with grid sizes  $dx \in [0.78, 0.39, 0.195]$ . The plot shows, for  $\Delta = 0.325$ , the full transient behaviour of the tip radius for all grid resolutions. In this case the results for all three grids is in good agreement. Higher under cooling makes the coarser grid less reliable and even  $dx = 0.39$  becomes less reliable.

## 5.1 Mesh Convergence

In order to gain further confidence in our computational approach we have undertaken a number of mesh convergence tests, in which we considered computational simulations in which the maximum level of mesh refinement is systematically increased. In order to appreciate the need for very fine grids at the phase interface Fig. 4 illustrates a cross section along the x-axis of a typical solution. This corresponds to the same parameter values as used to compute the dendrite illustrated in Fig. 1. It is clear that the phase variable,  $\phi$ , changes  $+1$  (bulk solid) to  $-1$  (bulk melt) over a very small distance, similarly the solute concentration varies rapidly both at, and immediately ahead of, the interface. The temperature variable,  $\theta$ , decays much more smoothly however – though a large domain is required to ensure that the boundary effect does not contaminate the solution. In addition to the interface width a further feature of significant interest is the geometry of the dendrite tip. Fig. 5 shows the computed evolution of the tip radius for three different undercoolings (0.325, 0.525 and 0.725) at  $Le = 40$ . The parameter,  $d_0 = 5\sqrt{2}/(8\lambda)$  is the (non-dimensional) capillary length as a function of the interface width. Our value for  $\lambda = 2$  gives,  $d_0 = .44$  indicating that the interface width in our simulation is of the order of the physical width. We use as our characteristic time scale,  $t_0 = 0.80(\tilde{d}_0^2\lambda^3)/\tilde{D}_c$  where  $\tilde{d}_0$  and  $\tilde{D}_c$  are the physically dimensioned capillary length and solute diffusivity coefficient respectively. Note that the results for Fig. 5 were computed using a fine mesh spacing of  $\Delta x = 0.39$ . It is an important question to ask if this is sufficiently fine for the solution to be insensitive to further mesh refinement.

Fig. 6 shows the computed tip radius as a function of time for three different maximum refinement levels ( $\Delta x = 0.78, 0.39$  and  $0.195$ ) for case  $Le = 40$  and  $\Delta = 0.325$ . The tip radius involves estimating a second derivative of  $\phi$  in the region where  $\phi = 0$ . This is undertaken by estimating the radius on the x-axis at  $\phi = 0$  using the phase field:

$$r = \left. \frac{\phi_{,x}}{\phi_{,uu}} \right|_{\phi(x,y,z)=0,y=0,z=0} \quad (74)$$

where  $\frac{\partial}{\partial u} = \frac{1}{\sqrt{2}} \left( \frac{\partial}{\partial y} + \frac{\partial}{\partial z} \right)$ ,  $\phi_{,x} \equiv \frac{\partial \phi}{\partial x}$  and  $\phi_{,uu} \equiv \frac{\partial^2 \phi}{\partial u^2}$  (as will become clear shortly, the  $u$  direction is more convenient than the  $y$  or  $z$  directions). Expression Eq. 74 comes from the definition

$$r = 1/\kappa \quad (75)$$

where the curvature,  $\kappa$ , is defined in the normalised direction  $\mathbf{u}$  to be

$$\kappa = \mathbf{u} \cdot \nabla \mathbf{n} \cdot \mathbf{u}. \quad (76)$$

On the x-axis, ignoring the  $z$  direction, and with a directional derivative in the  $y$  direction, we find, using  $\phi$  to compute



the normal  $\mathbf{n} \equiv \nabla\phi/|\nabla\phi|$ , that

$$\kappa = \frac{\partial n_2}{\partial y} = \frac{\partial}{\partial y} \frac{\phi_{,y}}{\sqrt{\phi_{,x}^2 + \phi_{,y}^2}} = \frac{\phi_{,yy}}{\phi_{,x}} \quad (77)$$

where, by symmetry  $\phi_{,xy}|_{y=0} = \phi_{,y}|_{y=0} = 0$  on the axis. Again, by symmetry, this relation holds for any normalised parameter and so

$$\kappa = \frac{\phi_{,yy}}{\phi_{,x}} = \frac{\phi_{,zz}}{\phi_{,x}} = \frac{\phi_{,uu}}{\phi_{,x}} \quad (78)$$

In practice, of course, the value  $\phi = 0$  lies between two successive nodes on the x-axis,  $i$  and  $i + 1$ . Furthermore, the x-axis lies, by definition, on  $y = z = 0$  where there are no grid points. We compute the derivatives only using the points

$$[i, 2, 2], [i + 1, 2, 2], [i, 3, 3], [i + 1, 3, 3] \quad (79)$$

which relates to the physical points

$$[i, j, k] \rightarrow [O_x, 0, 0] + \Delta x[i - 3/2, j - 3/2, k - 3/2] \quad (80)$$

where  $O_x$  is the x coordinate of the block origin. Thus, we know by symmetry, that the values at these points can be equated to the image nodes (which we do not use explicitly)

$$\begin{aligned} \phi_{[i,1,1]} &= \phi_{[i,2,2]}, \\ \phi_{[i,0,0]} &= \phi_{[i,3,3]}. \end{aligned} \quad (81)$$

We compute the radius of curvature using the direction  $u$  by

$$r = \left( \frac{\phi_x}{\phi_{uu}} + \frac{\phi}{\phi_x} \right) \Big|_{[i,2,2]} \quad (82)$$

where the last term is a correction to compensate for, in general,  $\phi \neq 0$  at the point  $[i, 2, 2]$ . This is discretised by

$$\begin{aligned} \phi_x &= \frac{\phi_{i+1,2,2} - \phi_{i-1,2,2}}{2\Delta x}, \\ \phi_{uu} &= \frac{-\phi_{i,2,2} + \phi_{i,3,3}}{2(\Delta x)^2} \end{aligned} \quad (83)$$

It is the nature of this approximation that leads to the oscillatory results that are observed on the least fine simulation ( $\Delta x = 0.78$ ). Nevertheless it is clear that the results for  $\Delta x = 0.39$  and  $\Delta x = 0.195$  are almost indistinguishable at the scale used here and so we have a good degree of confidence that a converged solution is obtained by  $\Delta x = 0.39$ . Similar computations of the large-time tip radius, on different levels of grid refinement, have been undertaken for two other cases, as shown in Tab. 2. Whilst the convergence is not so mature in every case, the evidence that results at  $\Delta x = 0.39$  are of sufficient accuracy to be of quantitative validity is very strong.

Note that the  $\Delta x = 0.195$  results for the other cases shown in Tab. 2 were not obtained by undertaking complete runs at this maximum refinement level. Instead, the large-time simulation computed using a maximum level of  $\Delta x = 0.39$  was restarted with a maximum level of  $\Delta x = 0.195$  and executed until a steady state tip radius was reached. This was tested for the ( $Le = 40, \Delta = 0.325$ ) case, and shown to give identical results. The approach is illustrated for  $Le = 100, \Delta = 0.325$  in Fig 7 and  $Le = 40, \Delta = 0.525$  in Fig. 8.

## 5.2 Numerical Performance

Having demonstrated the mesh convergence of our proposed technique in the previous subsection, we now consider the computational performance of the implementation used.

The most important feature of any successful multigrid implementation is that it should enable solutions of systems of algebraic equations to be obtained in a run time that is close to  $O(N)$  as  $N \rightarrow \infty$  where  $N$  is the number of degrees of freedom. In this simulation we begin with a small solid seed at the origin and this grows (under that right parameter conditions) in time. As it grows the region of maximum mesh refinement gets larger and larger, as the isosurface,  $\phi = 0$  has ever increasing area. This causes  $N$  to increase in time. Hence, an excellent test of our solver is whether the computed time to take each time step only grows in proportion to  $N$ . Unfortunately this test is harder to undertake than initially might be imagined since, as  $N$  increases the amount of memory required to compute a time step also

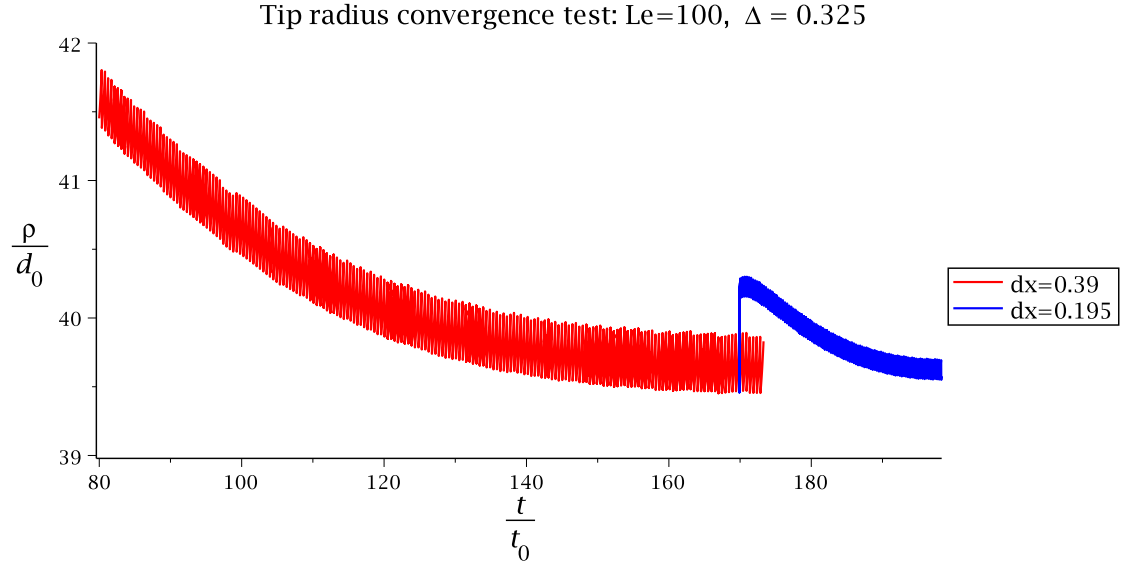


Figure 7: Convergence test on tip radius with grid sizes  $dx \in [0.39, 0.195]$ . A checkpoint file at  $dx = 0.39$  is used as an initial condition for a  $dx = 0.195$  to test convergence. The restart recovers from an initial transient before settling to a value very similar to the steady state at the coarser,  $dx = 0.39$  run.

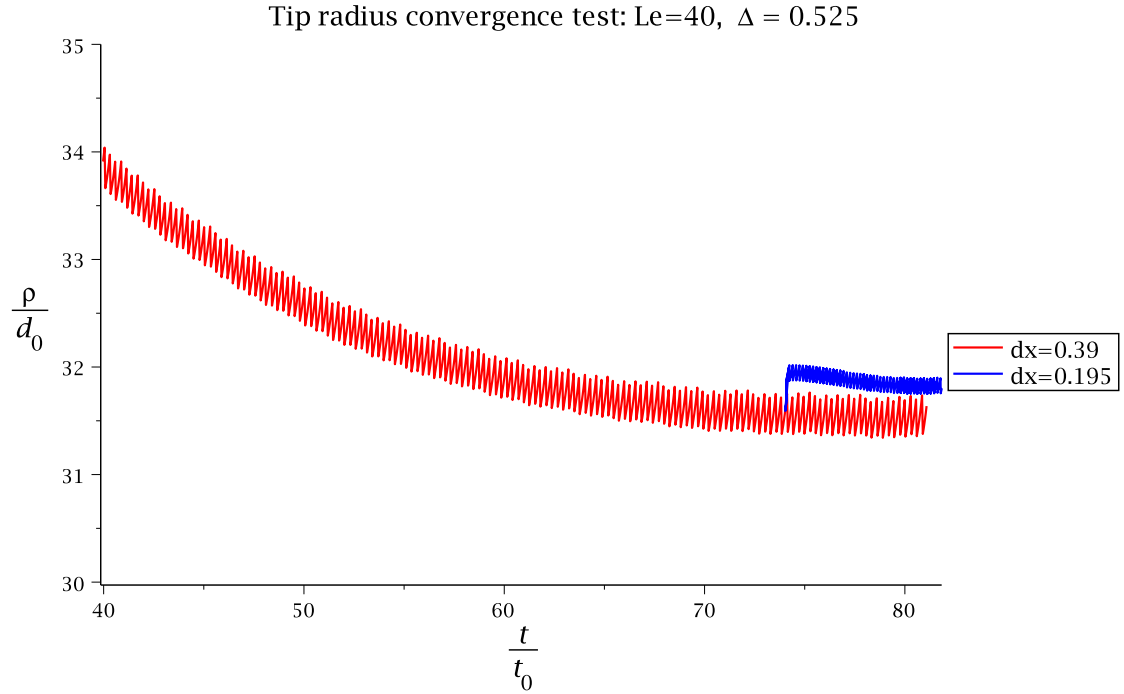


Figure 8: Convergence test on tip radius with grid sizes  $dx \in [0.39, 0.195]$ . A checkpoint file at  $dx = 0.39$  is used as an initial condition for a  $dx = 0.195$  to test convergence.

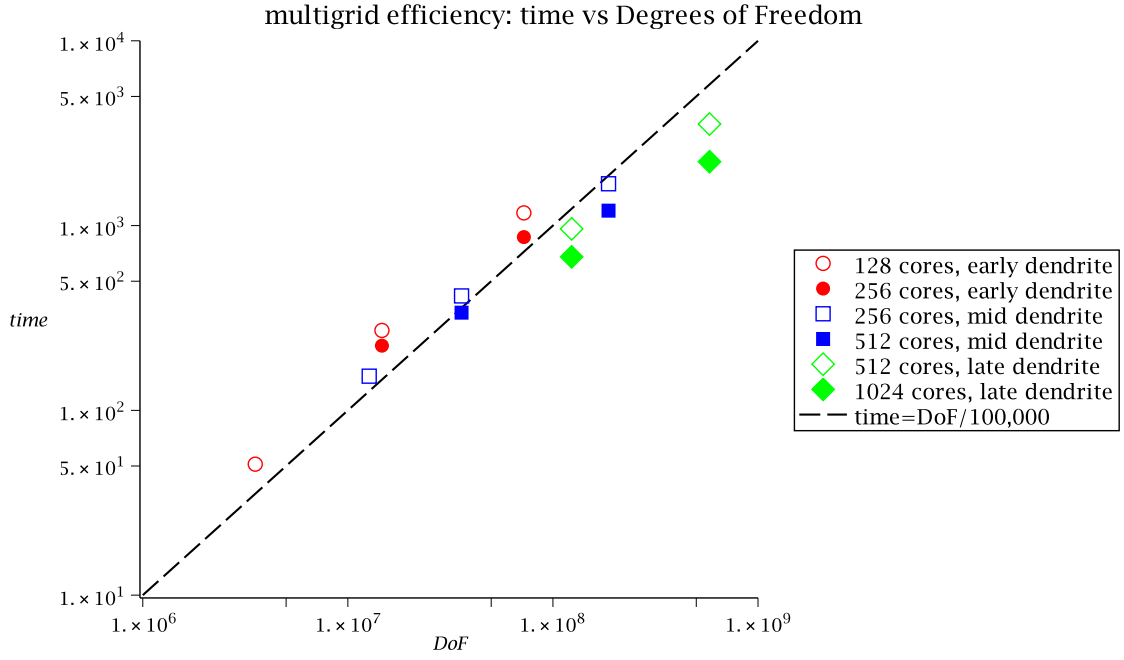


Figure 9: Multigrid efficiency. A demonstration of the linearity of solve time with the number of degrees of freedom. In all 6 cases the corresponding points fit well (for a single time step) to the line of slope 1.



Figure 10: Dendrite image:  $Le = 40, \theta = -0.525, t = 102, dx = 0.78$ . This simulation took 12 hours on a 12 core machine.

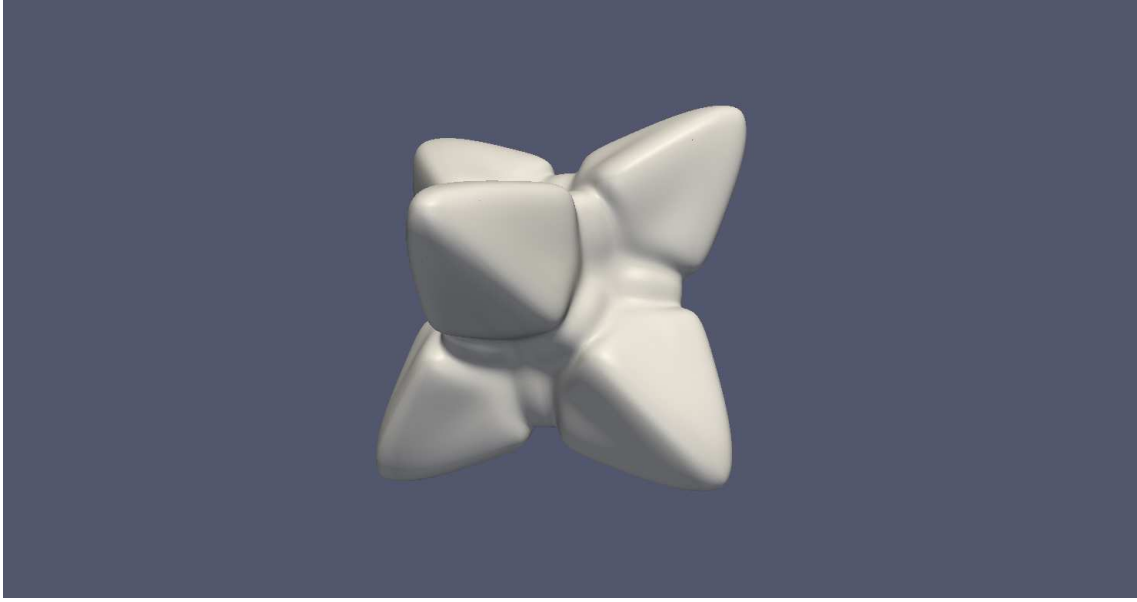


Figure 11: Dendrite image:  $Le = 40, \theta = -0.525, t = 186, \Delta x = 0.78$ . This simulation is of a dendrite with the coarsest maximum refinement and took about 40 hours to simulate on a 12 core machine. The last 10% of the run (a time interval of  $t = 18$ ) took about 10 hours.

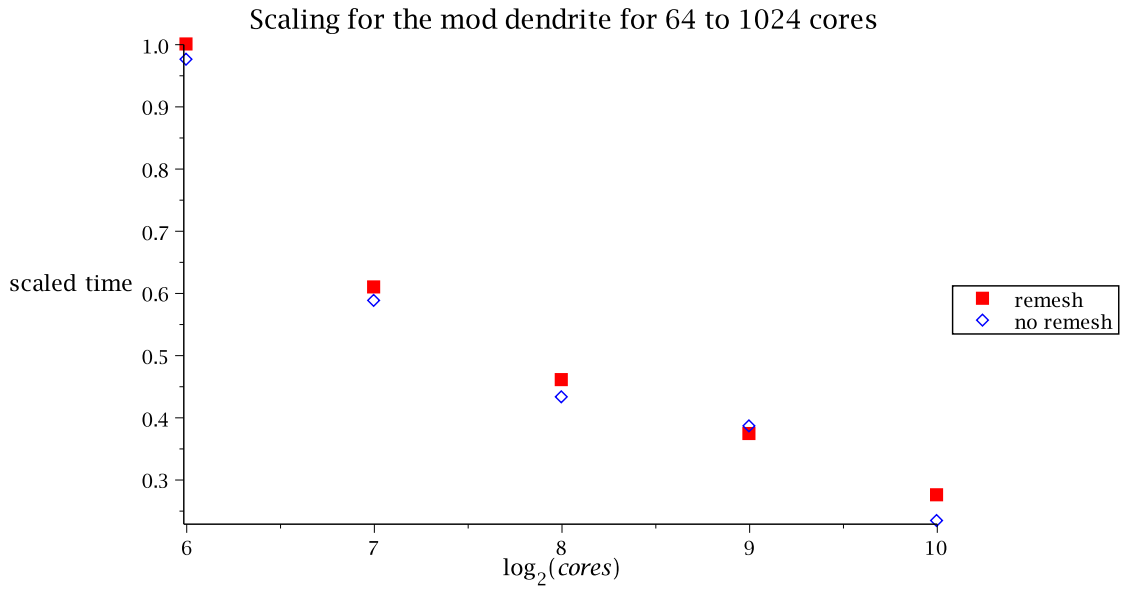


Figure 12: Plot of the relative wall-clock time to undertake 10 time steps at the mid dendrite stage (see Fig. 11), for mesh  $dx = 0.39$ , using different numbers of cores (64 to 1024).

Tip velocity vs time for  $Le=40$ ,  $\Delta = 0.39$

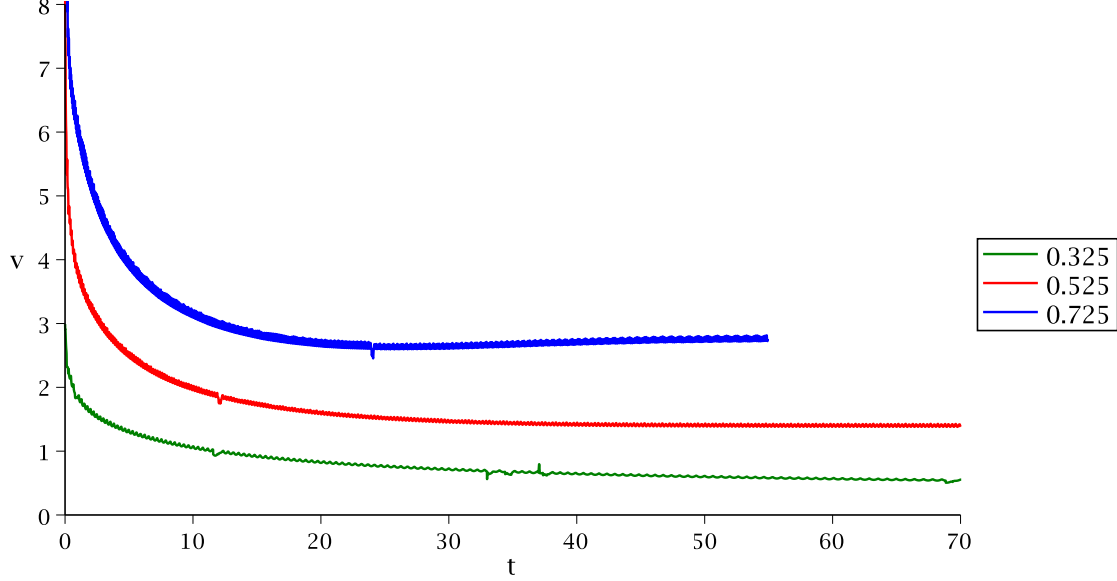


Figure 13: This is a companion plot to the tip radius plot 5: The tip velocity at  $dx = 0.39$  and  $Le = 40$  for a range of under cooling  $\Delta = 0.325, 0.525, 0.725$ . Though the tip radius has reached steady state in each of these cases the tip velocity for the highest undercooling cases is still increasing.

increases (linearly). As described in Sec 4 we deal with this through a distributed memory parallel implementation. We start the execution, with a small seed, using a modest number of cores (16 or 32 say) and increase these as the execution progresses. Furthermore, as discussed in the previous subsection, we also wish to consider different choices for the maximum level of mesh refinement, which also impacts on the number of computational nodes to be used and therefore the total memory requirement.

Fig 9 shows a selection of timings for a set of computations for a single, representative, test case ( $Le = 40, \Delta = 0.525$ ) using different cores. The vertical axis shows the execution time for a single implicit time step and the horizontal axis shows the number of degrees of freedom (both on log scales). Timings are taken just as the dendrite is starting to form (early dendrite see Fig. 10), part way through its formation (mid dendrite Fig. 11), and once the dendrite is clearly formed (late dendrite Fig. 1). Timings are also taken using different maximum refinement levels (either 2 or 3 per case). It is very clear from Fig. 9 that multigrid performance is achieved throughout these different stages of the evolution of the dendrite and at different maximum refinement levels. This can be seen from the excellent proximity of the points to the time line of slope one ( $t = DoF/100,000$ ) on the log-log scale.

Note that our use of distributed memory parallel computing throughout this work has been aimed primarily at providing the capacity to solve large systems (up to and beyond a billion degrees of freedom per implicit time step) in a computationally efficient manner. It is clear from Fig. 12 that the strong parallel scalability of our implementation is not optimal. Nevertheless it is apparent that, as well as providing additional memory capacity to allow larger problems to be tackled, our parallel implementation also continues to improve the speed of the execution each time each time the core count is increased so long as the number of degrees of freedom is sufficiently large.

## 6 Conclusions

We have presented, in detail, the mathematical model and methods employed to simulate, for the first time, a three dimensional, fully coupled thermal-solute-phase field model for dendritic growth. This was achieved through the coupling of multiple numerical techniques from compact discrete finite difference stencils, AMR, MLAT and parallel execution.

For moderate Lewis numbers we have been able to obtain results at sufficient grid resolution, which we have confidence can provide quantitative accuracy in 3-d for the first time. This break through into three dimensional fully coupled thermal simulation is of importance, since heat generation at the growing two dimensional surface is very much an integral part of the natural physical process and two dimensional results therefore have little quantitative value. Our next goal is to undertake a systematic simulation of the effect of increasing Lewis number.

We end with the observation that, though, tip radius is fully converged, the tip velocity is only near steady state. We give the companion plot to Fig. 5 for the tip velocities for the same parameters in Fig. 13.

## 7 Acknowledgements

This research was funded by EPSRC grant number EP/H048685. We are also grateful for the use of the HECToR UK National Supercomputing Service.

## References

- [1] Alain Karma. Phase-field formulation for quantitative modeling of alloy solidification. *Phys. Rev. Lett.*, 87:115701, Aug 2001.
- [2] J. Rosam, P. K. Jimack, and A. M. Mullis. Quantitative phase-field modeling of solidification at high lewis number. *Phys. Rev. E*, 79:030601, Mar 2009.
- [3] A. M. Mullis, C. E. Goodyer, and P. K. Jimack. Towards a Three-Dimensional Phase-Field Model of Dendritic Solidification with Physically Realistic Interface Width. *Transactions of the Indian Institute of Metals*, 65(6):617–621, 2012.
- [4] Alain Karma and Wouter-Jan Rappel. Phase-field simulation of three-dimensional dendrites: is microscopic solvability theory correct? *Journal of Crystal Growth*, 174(14):54 – 64, 1997. American Crystal Growth 1996 and Vapor Growth and Epitaxy 1996.
- [5] J. C. Ramirez, C. Beckermann, A. Karma, and H-J J. Diepers. Phase-field modeling of binary alloy solidification with coupled heat and solute diffusion. *Physical review. E, Statistical, nonlinear, and soft matter physics*, 69(5 Pt 1), May 2004.
- [6] Samuel M. Allen and John W. Cahn. A microscopic theory for antiphase boundary motion and its application to antiphase domain coarsening. *Acta Metallurgica*, 27(6):1085 – 1095, 1979.
- [7] John W Cahn. On spinodal decomposition. *Acta Metallurgica*, 9(9):795 – 801, 1961.
- [8] Peter MacNeice, Kevin M. Olson, Clark Mobarry, Rosalinda de Fainchtein, and Charles Packer. Paramesh: A parallel adaptive mesh refinement community toolkit. *Computer Physics Communications*, 126(3):330 – 354, 2000.
- [9] K. Olson. Paramesh: A parallel adaptive grid tool. in parallel computational fluid dynamics 2005: Theory and applications. *ed A. Deane et al. (Elsevier)*, 2006.
- [10] A high-order compact formulation for the 3d poisson equation. *Numer. Methods Partial Differential Eq.*, 12:235 243, 1996.
- [11] J. Rosam, P.K. Jimack, and A.M. Mullis. An adaptive, fully implicit multigrid phase-field model for the quantitative simulation of non-isothermal binary alloy solidification. *Acta Materialia*, 56(17):4559 – 4569, 2008.
- [12] A. Brandt. Multi-level adaptive solutions to boundary-value problems. *Math. Comp.*, 31:333 – 390, 1977.
- [13] P.K. Jimack J. Rosam and A.M. Mullis. A fully implicit fully adaptive time and space discretization method for phase-field simulation of binary alloy solidification. *J. Comput. Phys.*, vol.225:1271 – 1287, 2007.
- [14] A.M. Mullis J. Green, P.K. Jimack and J. Rosam. An adaptive, multilevel scheme for the implicit solution of three-dimensional phase-field equations. *Numerical Methods for PDEs*, 27:106 – 120, 2011.
- [15] Jun-Ho Jeong, Nigel Goldenfeld, and Jonathan A. Dantzig. Phase field model for three-dimensional dendritic growth with fluid flow. *Phys. Rev. E*, 64:041602, Sep 2001.
- [16] N. Goldenfeld N. Provatas and J.A. Dantzig. Adaptive mesh refinement computation of solidification microstructures using dynamic data structures. *J. Comput. Phys.*, 148:265 – 290, 1999.
- [17] A.M. Mullis H. Dong Y. Xie C.E. Goodyer, P.K. Jimack. On the fully implicit solution of a phase-field model for binary alloy solidification in three dimensions. *Advances in App. Math. and Mech*, 4, 2012.